Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Desarrollo de una interfaz de usuario para programas de inversión LIDAR: despolarización por aerosoles

TITULACIÓN: Grado en Ingeniería de Sistemas de Telecomunicación

AUTOR: Álvaro Galdaba Toledano

DIRECTOR: Alejandro Rodríguez Gómez

FECHA: 23 de octubre del 2018

Título: Desarrollo de una interfaz de usuario para programas de inversión LIDAR: despolarización por aerosoles

Autor: Álvaro Galdaba Toledano

Director: Alejandro Rodríguez Gómez

Fecha: 23 de octubre del 2018

Resumen

El proyecto que se ha desarrollado para este trabajo de final de grado tiene como objetivo la integración de una serie de rutinas en MATLAB, que permiten obtener información sobre la despolarización que los aerosoles atmosféricos producen sobre la señal láser producida por un LIDAR multi-longitud de onda.

Para llevar a cabo el trabajo, se ha buscado información sobre cómo se puede crear una interfaz gráfica utilizando MATLAB, su funcionamiento interno y propiedades, además de todo lo relacionado para intercambiar datos entre funciones y representación de los mismos.

Los programas originales han sido adaptados para poder trabajar correctamente con las interfaces de usuario. Cada operación que se llevaba a cabo por medio de comandos se ha programado de manera que pueda utilizarse de forma más amigable e intuitiva, mediante elementos gráficos como pueden ser botones, cuadros de texto y gráficas para representación de datos.

Cada programa, carga una serie de archivos que contienen los datos que se utilizan internamente para la obtención y representación de resultados. Dependiendo del tipo de archivo cargado la aplicación procesa los datos de una forma u otra, es por esto que se ha hecho una pequeña explicación del contenido que tiene cada tipo de archivo y si se tratan de archivos brutos o procesados.

El resultado final consiste en un menú principal llamado *Depol Menu*, que contiene las GUI de *Calibration* y *Analysis* donde están adaptadas las rutinas utilizadas por los programas originales.

El presente trabajo, aparte del desarrollo de la aplicación y la comprobación del correcto funcionamiento de la misma, también tiene documentado una breve introducción de cómo funciona un sistema LIDAR, así como el sistema que tiene montado la Universidad Politécnica de Cataluña (Campus Nord) para obtener las medidas que se utilizan por la interfaz de usuario desarrollada.

Títol: Desenvolupament d'una interfície d'usuari per a un programari d'inversió LIDAR: despolarització per aerosols

Autor: Álvaro Galdaba Toledano

Director: Alejandro Rodríguez Gómez

Data: 23 d'octubre del 2018

Resum

El projecte que s'ha desenvolupat per a aquest treball de final de grau té com a objectiu la integració d'una sèrie de rutines en MATLAB, que permeten obtenir informació sobre la despolarització que els aerosols atmosfèrics produeixen sobre el senyal làser produïda per un LIDAR multi-longitud d'ona.

Per dur a terme el treball, s'ha buscat informació sobre com es pot crear una interfície gràfica utilitzant MATLAB, el seu funcionament intern i propietats, a més de tot el relacionat per intercanviar dades entre funcions i representació dels mateixos.

Els programes originals han estat adaptats per poder treballar correctament amb les interfícies d'usuari. Cada operació que es duia a terme per mitjà de comandes s'ha programat de manera que pugui utilitzar-se de manera més amigable i intuïtiva, mitjançant elements gràfics com poden ser botons, quadres de text i gràfiques per a representació de dades.

Cada programa, carrega una sèrie d'arxius que contenen les dades que s'utilitzen internament per a l'obtenció i representació de resultats. Depenent del tipus d'arxiu carregat l'aplicació processa les dades d'una manera o una altra, és per això que s'ha fet una petita explicació del contingut que té cada tipus d'arxiu i si es tracten d'arxius bruts o processats.

El resultat final consisteix en un menú principal anomenat *Depol Menu*, que conté les GUI de *Calibration* i *Analysis* on estan adaptades les rutines utilitzades pels programes originals.

El present treball, a part del desenvolupament de l'aplicació i la comprovació del correcte funcionament de la mateixa, també té documentat una breu introducció de com funciona un sistema LIDAR, així com el sistema que té muntat la Universitat Politècnica de Catalunya (Campus Nord) per obtenir les mesures que s'utilitzen per la interfície d'usuari desenvolupada.

Title: Development of a user interface for LIDAR inversion programs: depolarization by aerosols

Author: Álvaro Galdaba Toledano

Director: Alejandro Rodríguez Gómez

Date: October, 23th 2018

Overview

The project that has been developed for this work at the end of degree aims to integrate a series of routines in MATLAB, which allow us to obtain information about the depolarization that atmospheric aerosols produce on the laser signal produced by a multi-wavelength LIDAR.

In order to carry out the work, information has been looked for information on how a graphic interface can be created using MATLAB, its internal working and properties, in addition to all the related to exchange data between functions and their representation.

The original programs have been adapted to be able to work correctly with the user interfaces. Each operation carried out by means of commands has been programmed so that it can be used in a more friendly and intuitive way, using graphic elements such as buttons, text boxes and graphics for data representation.

Each program loads a series of files that contain the data that is used internally to obtain and represent results. Depending of the type of file loaded the application processes the data in one way or another, this is why a small explanation has been made of the content that each type of file has and whether they are raw or processed files.

The final result consists of a main menu called *Depol Menu*, which contains the *Calibration* and *Analysis* GUI where the routines used by the original programs are adapted.

The present work, apart from the development of the application and the verification of the correct working of it, also has documented a brief introduction of how a LIDAR system works, as well as the system that the Universitat Politècnica de Catalunya (Campus Nord) has mounted for obtain the measurements that are used by the user interface developed.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN Y OBJETIVOS

Los objetivos de este proyecto consisten en adaptar una serie de programas, poco amigables de cara al usuario, que permiten obtener información sobre la despolarización de aerosoles, a partir de medidas de un LIDAR (radar láser [1]) a una interfaz gráfica realizada con MATLAB. Se debe comprobar que todas las funciones del programa se comuniquen para poder realizar los cálculos y presentarlos correctamente por pantalla. Otro punto muy importante, es conseguir que la GUI desarrollada sea robusta y no deje de funcionar cuando haya un imprevisto durante la ejecución de la aplicación. Finalmente, hay que comprobar que los resultados obtenidos sean iguales a los obtenidos con el software original.

La interfaz gráfica desarrollada para este proyecto está formada por un menú principal que contiene dos opciones para procesar las medidas. Primero tenemos la interfaz de calibración, que se encarga de procesar las medidas para obtener una serie de parámetros necesarios para la otra opción de la interfaz, el análisis de medidas, donde se calcula la despolarización de volumen y de partículas.

En el primer capítulo de esta memoria se define el concepto del LIDAR, cómo funciona y qué se puede obtener utilizando este sistema. Además, se explican las diferentes fórmulas que se utilizan durante la ejecución de las aplicaciones para obtener los diferentes resultados que se deben presentar al usuario.

En el segundo capítulo se presenta el concepto de GUI y el entorno software con el que se ha trabajado para su creación. También se ha profundizado en las características más importantes cuando se crea una interfaz gráfica y los archivos generados al guardar el proyecto por primera vez. Por otro lado, se explica la forma de pasar datos entre funciones, la presentación de datos por pantalla mediante gráficas y una de las funciones que contiene toda GUI por defecto.

El tercer capítulo describe los diferentes tipos de archivos de datos que utiliza el software desarrollado, así como el contenido de cada uno de ellos y que se utilizarán a la hora de realizar los cálculos.

En el cuarto capítulo se detalla el funcionamiento de las diferentes GUI que forman este proyecto, las diferentes opciones que se pueden encontrar para cada aplicación y los archivos generados cuando se guardan los resultados.

Finalmente, se analiza críticamente si se han llevado a cabo los objetivos fijados en un principio y si se podrían incluir posibles mejoras en la interfaz gráfica desarrollada.

CAPÍTULO 1. LIDAR

En este primer capítulo se presenta y explica el concepto de LIDAR, canal de despolarización y la formulación que será utilizada por los programas. Finalmente se hará una breve introducción al sistema montando por la UPC para la obtención de medidas.

1.1. Sistema LIDAR

Un sistema LIDAR [1] emite de manera constante pulsos de luz, mediante un láser, hacia la atmosfera. Estos pulsos son retrodispersados por la atmósfera y recogidos por la apertura de un telescopio.

De esta manera, la luz captada por el telescopio se detecta, procesa y se digitaliza para analizar y obtener información sobre las diferentes capas de partículas que se pueden encontrar en la atmósfera.

Otros sistemas que utilizan los mismos principios, pero no utilizan la luz para detectar objetos o partículas pueden ser: RADAR donde se emiten ondas radio y SONAR donde se emiten ondas acústicas.

Los LIDAR pueden trabajar con distintas longitudes de onda dependiendo del láser que se utilice para emitir y los filtros utilizados en el sistema de detección.

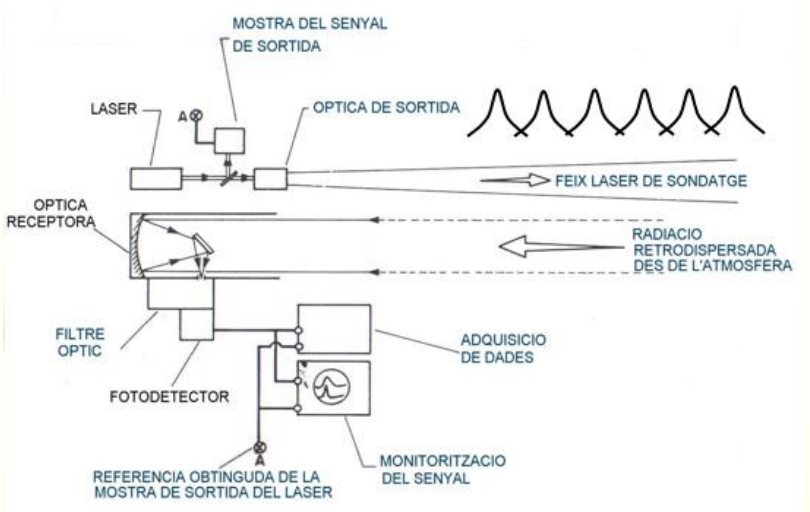

Fig. 1.1 Esquema LIDAR [2].

La luz transmitida por el láser entra en contacto con las diferentes capas de aerosoles que se encuentran en la atmosfera, esto provoca que los fotones se dispersen. Durante un período de tiempo y con la ayuda de un telescopio se capturan los fotones rebotados, que son procesados hasta obtener una nube de

puntos como el de la figura 1.2, se puede observar el coeficiente de retrodispersión (β) para diferentes alturas de capas de aerosoles.

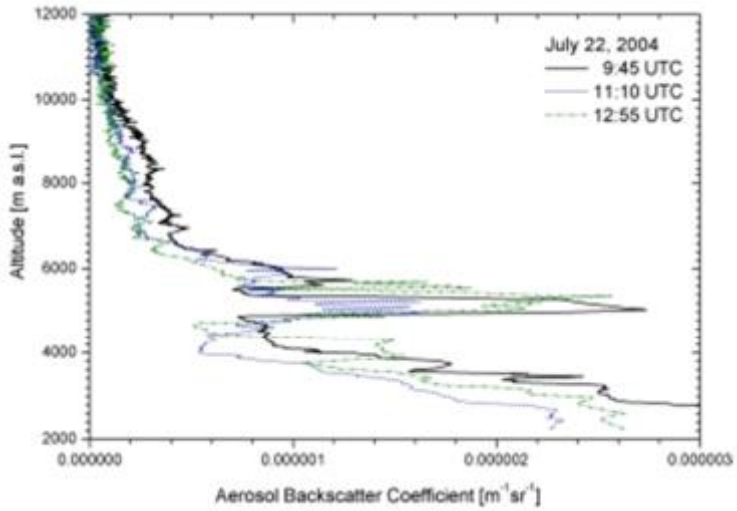

Fig. 1.2 Capas atmósfera [35].

Ecuación del LIDAR elástico [1] viene definida por:

$$P_R(R) = \frac{P_o A_R}{R^2} \cdot \Delta R \cdot \beta(R) e^{-2\int_0^R \alpha(x)dx} \quad \textbf{(1.1)}$$

Donde se puede aproximar: $\Delta R = \frac{c\tau}{2}$

Finalmente, si se sustituye ΔR en la ecuación **1.1** se obtiene:

$$P_R(R) = \frac{P_o A_R}{R^2} \cdot \frac{c\tau}{2} \cdot \beta(R) e^{-2\int_0^R \alpha(x)dx} \quad \textbf{(1.2)}$$

Donde P_R es la potencia recibida por el telescopio desde una distancia R, P_o la potencia máxima del láser durante la duración del pulso τ , A_R es el área efectiva del telescopio y c velocidad de la luz.

La información sobre la atmósfera está reflejada en los coeficientes ópticos $\beta(R)$ y $\alpha(R)$, están presentes en los aerosoles en suspensión y dependen de la variable R ya que el valor de estos coeficientes es diferente según la distancia. El coeficiente β representa la retrodispersion de la luz que se dispersa en dirección a 180º del ángulo de incidencia, por otro lado el coeficiente α representa la absorción y dispersión (atenuación).

La distancia R es el recorrido que hace un pulso en ir y volver en un periodo de tiempo t, por lo tanto se puede calcular esta distancia a partir de la siguiente fórmula:

$$R = \frac{ct}{2} \cdot \alpha(R) \tag{1.3}$$

Para el caso del LIDAR elástico los fotones tienen la misma longitud de onda en emisión y recepción (una vez retrodispersados). En consecuencia, estos fotones tienen la misma energía, ya que matemáticamente la energía del fotón [3] se puede calcular como:

$$E = \frac{hc}{\lambda} \tag{1.4}$$

Donde E es la energía del fotón, h la constante de Planck, c la velocidad de la luz y λ la longitud de onda. Al ser h y c constantes fijas, la energía depende inversamente de la longitud de onda seleccionada.

1.2. Canales de despolarización

El sistema LIDAR de la UPC añade un telescopio auxiliar para detección de despolarización [35]. El telescopio principal recoge la luz retrodispersada a 532 nm, sin tener en cuenta la polarización. El telescopio auxiliar detecta la despolarización de la luz retrodispersada en la misma longitud de onda.

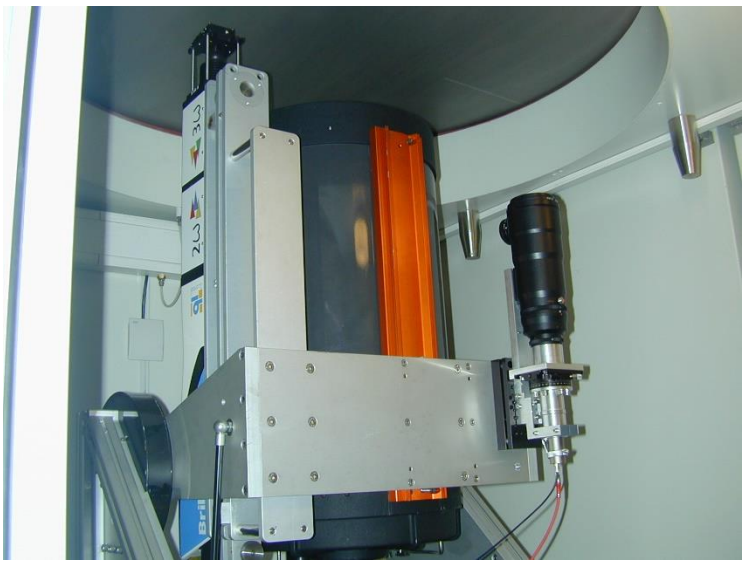

Fig. 1.3 Sistema LIDAR UPC.

En la figura 1.3, se puede observar LIDAR de la UPC: en la parte izquierda se encuentra el láser, en medio el telescopio principal y en la derecha el canal de despolarización auxiliar.

El canal de despolarización auxiliar tiene la siguiente arquitectura:

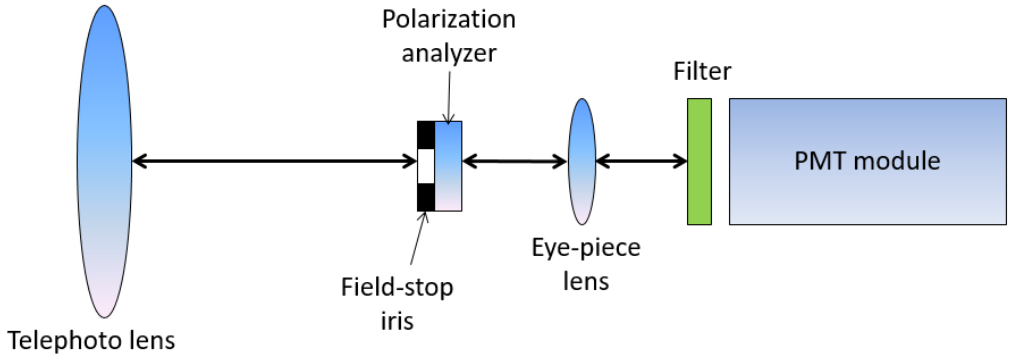

Fig. 1.4 Componentes canal despolarización auxiliar.

Este sistema permite calcular la relación de despolarización de volumen comparando las salidas de los dos canales. Por esta razón, el polarizador del canal de despolarización auxiliar debe estar orientado a 90° de la polarización lineal transmitida.

Adicionalmente en 2018 se ha instalado un canal adicional que permite medir la despolarización de la luz retrodispersada a 355 nm [4].

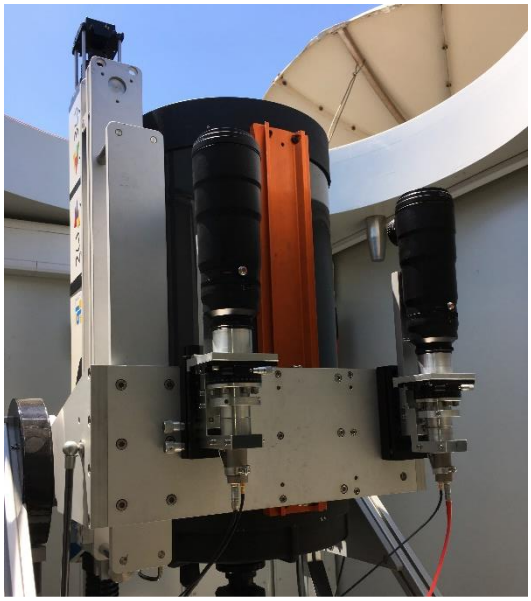

Fig. 1.5 Configuración LIDAR UPC con los dos canales de despolaización.

Para calcular el valor real del polarizador, se compara la señal del canal de despolarización con el canal de potencia total en las dos posiciones de

calibración (+45° y -45° de la posición nominal). Todo el desarrollo matemático que se refiere a continuación puede encontrarse en las referencias [5] y [6].

La señal de voltaje obtenida a la salida del detector de canal de potencia total se puede definir como:

$$S_T(R) = V_T(R) \cdot P_T(R) \quad \textbf{(1.5)}$$

Donde $V_T(R)$ es la responsividad del canal de potencia total para una distancia R, incluyendo los efectos de la superposición parcial, la trasmisión por fibras, las perdidas en la unidad de separación de longitud de onda (PMT) y el detector de responsividad.

Por otro lado, $P_T(R)$ es la potencia de la luz retrodispersada y que es acumulada por el telescopio principal en la longitud de onda que se está trabajando. Está formada por la suma de las componentes co-polar ($P_{\parallel}(R)$) y cross-polar ($P_{\perp}(R)$) de la potencia recibida.

La señal de voltaje obtenida a la salida del detector de canal de despolarización (auxiliar) se escribe como:

$$S_{Dep}(\varphi, R) = V_{Dep}(R) \cdot P_{\varphi}(R) \quad \textbf{(1.6)}$$

Donde $V_{Dep}(R)$ es la responsividad del canal de despolarización para una distancia R, incluyendo los efectos de la superposición parcial. Si el sistema está formado por dos telescopios, $V_{Dep}(R)$ es diferente de $V_T(R)$ a causa de las diferentes propiedades ópticas y posiciones de los telescopios.

Por otro lado, $P_{\varphi}(R)$ es la parte de la luz retrodispersada y despolarizada a distancia R, es detectada cuando el polarizador gira un ángulo φ respecto al plano de polarización de los pulsos transmitidos. En consecuencia, interviene la luz de retrodispersión co-polar y cross-polar dependiendo del ángulo φ:

$$P_{\varphi}(R) = P_{\parallel}(R)cos^2\varphi + P_{\perp}(R)sin^2\varphi \quad \textbf{(1.7)}$$

La función del sistema del canal de despolarización viene definida como:

$$V^*(R) = \frac{V_{Dep}(R)}{V_T(R)} \quad \textbf{(1.8)}$$

Debido a la dificultad que existe para calcular los valores de $V_{Dep}(R)$ y $V_T(R)$, es necesario utilizar otro método para estimar $V^*(R)$. Para calcular el valor de $V^*(R)$ se necesita un proceso de calibración donde se comparan las señales de salida del canal de despolarización y de potencia total.

El observable δ^* se puede expresar como:

$$\delta^*(\varphi, R) = \frac{S_{Dep}(\varphi, R)}{S_T(R)} = \frac{V_{Dep}(R) \cdot [P_{\parallel}(R)\, cos^2\varphi + P_{\perp}(R)\, sin^2\varphi]}{V_T(R) \cdot [P_{\parallel}(R) + P_{\perp}(R)]} \quad \textbf{(1.9)}$$

Por otro lado, se determina la despolarización de volumen lineal producida por la atmosfera [7] como el cociente entre las componentes de potencia:

$$\delta^V(R) = \frac{P_{\perp}(R)}{P_{\parallel}(R)} \quad \textbf{(1.10)}$$

Las componentes $P_{\perp}(R)$ y $P_{\parallel}(R)$ son iguales para un ángulo $\varphi = 90°$, teniendo únicamente luz retrodipersada y despolarizada.

Se puede relacionar el observable $\delta^*(\varphi, R)$ y la despolarización de volumen lineal $\delta^V(R)$ si se divide el numerador y denominador por la componente $P_{\parallel}(R)$ en la ecuación **1.9**:

$$\delta^*(\varphi, R) = V^*(R) \cdot \frac{cos^2\varphi + \delta^V(R)\, sin^2\varphi}{1 + \delta^V(R)} \quad \textbf{(1.11)}$$

Para calcular $V^*(R)$, se realizan dos medidas donde se utilizan dos valores para el ángulo φ, separados $90°$ con respecto a la posición nominal φ_0:

$$\begin{aligned} \varphi_- &= \varphi_0 - 45^{\circ} \\ \varphi_+ &= \varphi_0 + 45^{\circ} \end{aligned} \quad \textbf{(1.12)}$$

Actualizando la ecuación **1.11** se obtiene:

$$\begin{aligned} \delta^*(\varphi_-, R) &= V^*(R) \cdot \frac{cos^2\varphi_- + \delta^V(R)\, sin^2\varphi_-}{1 + \delta^V(R)} \\ \delta^*(\varphi_+, R) &= V^*(R) \cdot \frac{cos^2\varphi_+ + \delta^V(R)\, sin^2\varphi_+}{1 + \delta^V(R)} \end{aligned} \quad \textbf{(1.13)}$$

Independientemente del valor de φ_0 se puede obtener:

$$V^*(R) = \delta^*(\varphi_-, R) + \delta^*(\varphi_+, R) \tag{1.14}$$

Al realizar las medidas de despolarización se orienta el polarizador a una posición donde $\varphi = \varphi_0$, que teóricamente debería ser $90°$.

Mediante una configuración manual se puede asegurar que la calibración del canal de despolarización se realiza en dos posiciones diferentes, donde la diferencia $\varphi_+ - \varphi_- = 90°$ contiene errores por debajo de $\pm 0.1°$. La posición φ_0 es propensa a errores grandes, ya que puede ser afectada por alguna inestabilidad a la hora de trasmitir el pulso o algún error mecánico.

Para calcular el valor real de φ_0 se restan las ecuaciones **1.13**:

$$\begin{aligned} \delta^*(\varphi_-, R) - \delta^*(\varphi_+, R) \quad &= V^*(R) \cdot \frac{1 - \delta^V(R)}{1 + \delta^V(R)} \cdot \cos 2\varphi_- \\ &= V^*(R) \cdot \frac{1 - \delta^V(R)}{1 + \delta^V(R)} \cdot \sin 2\varphi_0 \\ &= [\delta^*(\varphi_-, R) + \delta^*(\varphi_+, R)] \cdot \frac{1 - \delta^V(R)}{1 + \delta^V(R)} \cdot \sin 2\varphi_0 \end{aligned} \tag{1.15}$$

Behrendt y Nakamura [8] calcularon $\delta^V(R)$ para la atmósfera molecular, es decir, la despolarización de la luz retrodispersada solo por moléculas. Este valor, llamado δ_m^V, depende del ancho de banda del filtro utilizado en el canal de despolarización.

La calibración del valor actual de φ_0 puede realizarse si se selecciona una parte de la señal que proviene de la atmósfera libre de aerosoles para una distancia R_{mol}, de modo que $\delta^V = \delta_m^V$:

$$\begin{aligned} \sin 2\varphi_0 &= \frac{1 + \delta_m^V}{1 - \delta_m^V} \cdot \frac{\delta^*(\varphi_-, R_{mol}) - \delta^*(\varphi_+, R_{mol})}{\delta^*(\varphi_-, R_{mol}) + \delta^*(\varphi_+, R_{mol})} \\ \sin 2\varphi_0 &\approx \frac{\delta^*(\varphi_-, R_{mol}) - \delta^*(\varphi_+, R_{mol})}{\delta^*(\varphi_-, R_{mol}) + \delta^*(\varphi_+, R_{mol})} \end{aligned} \tag{1.16}$$

Para δ_m^V se utiliza el valor $3.8 \cdot 10^{-3}$ calculado por Behrendt y Nakamura [8] para 532 nm y de $8 \cdot 10^{-3}$, calculado por Alejandro Rodríguez [4]. Los valores de $\delta^*(\varphi_+, R_{mol})$ y $\delta^*(\varphi_-, R_{mol})$ se espera que sean muy pequeños, por ello para estimar φ_0 se usaran mediciones nocturnas promediadas cada 150 minutos.

Si se conoce el valor actual de φ_0, podemos calcular la relación de despolarización de volumen de cualquier medida despejando el factor $\delta^V(R)$ en la ecuación **1.11**:

$$\delta^V(R) = \frac{\delta^*(\varphi_0, R) - V^*(R) \cdot cos^2\varphi_0}{V^*(R) \cdot sin^2\varphi_0 - \delta^*(\varphi_0, R)} \quad \textbf{(1.17)}$$

Finalmente, la relación de despolarización de partícula se puede calcular combinando la relación de despolarización de volumen con los perfiles de retrodispersión molecular (β^m) y de aerosoles (β^{p}).

$$\delta^{\mathrm{p}}(R) = \frac{[1 + \delta_m^V] \cdot \delta^V(R) \cdot \rho(R) - [1 + \delta^V(R)] \cdot \delta_m^V}{[1 + \delta_m^V] \cdot \rho(R) - [1 + \delta^V(R)]} \quad \textbf{(1.18)}$$

Donde $\rho(R)$ es la relación de retrodispersión de los aerosoles:

$$\rho(R) = \frac{\beta^m(R) + \beta^{\mathrm{p}}(R)}{\beta^m(R)} \quad \textbf{(1.19)}$$

El valor de los perfiles de retrodispersión molecular y de aerosol se calcula mediante un algoritmo Klett–Fernald [10,11] (KFS) o una inversión Raman [12,13] realizada sobre la señal de potencia total del canal, procedimientos que ya han sido implementados con anterioridad y que quedan por tanto fuera de los objetivos de este TFG.

CAPÍTULO 2. INTRODUCCIÓN GUI MATLAB

Una vez introducida la teoría, se describirá el concepto de GUI, como se puede crear y los archivos generados durante este proceso. También se detalla cómo se intercambian los datos entre diferentes funciones y la forma de presentarlos por pantalla.

2.1. Qué es una GUI?

Las interfaces gráficas de usuario [15] o GUI (Graphical User Interface) permiten la comunicación entre el usuario y el sistema operativo de un ordenador, a través de un entorno visual, intuitivo y amigable.

Permite al usuario controlar un programa sin la necesidad de aprender el lenguaje de programación que hay detrás de la aplicación. No es necesario ejecutar comandos para su funcionamiento, sino que el usuario debe interactuar con el programa a través de diferentes elementos gráficos que contienen la programación de la aplicación.

A la hora de crear interfaces gráficas, existen múltiples lenguajes de programación como Visual Basic, C# y C. Para este proyecto utilizaremos MATLAB.

Para el diseño del programa se usa la herramienta de MATLAB llamada GUIDE [9].

2.2. Creación de una GUI

Para poder implementar una GUI se utiliza el editor de interfaces de usuario GUIDE (Graphical User Interface Development Enviroment) de MATLAB. Esta herramienta permite al usuario diseñar la interfaz gráfica de una manera rápida y fácil.

Por defecto, cuando se selecciona un componente se genera automáticamente una función vacía. Por ello, para que cada elemento seleccionado realice la acción deseada hay que modificar las funciones que se van generando añadiendo el código correspondiente.

2.2.1. Herramienta GUIDE

Una vez nos encontramos con MATLAB abierto tenemos dos opciones para poder empezar a utilizar GUIDE. Para ello se deben seguir las siguientes instrucciones:

HOME → New → App →GUIDE

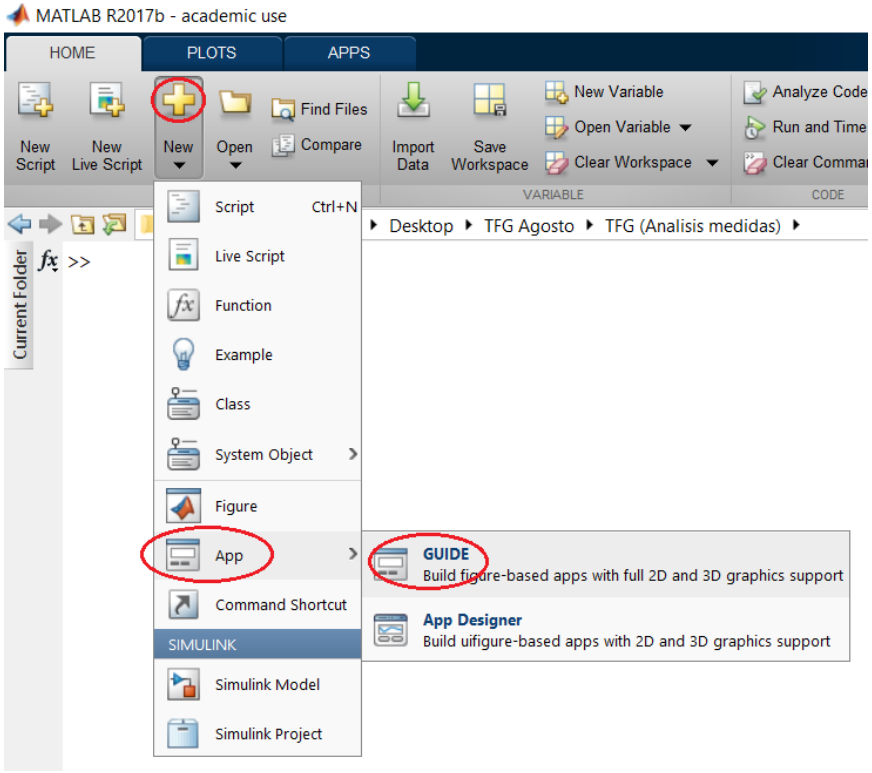

Fig. 2.1 Pasos para iniciar GUIDE.

La otra alternativa, es escribir directamente el comando *guide* en la ventana de comandos de MATLAB:

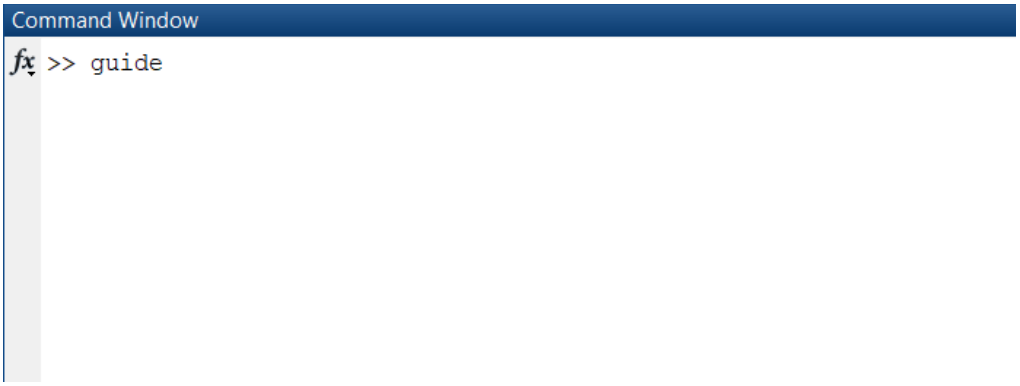

Fig. 2.2 Iniciando GUIDE con ventana de comandos.

Una vez iniciado GUIDE se muestra la pantalla de inicio de la siguiente figura:

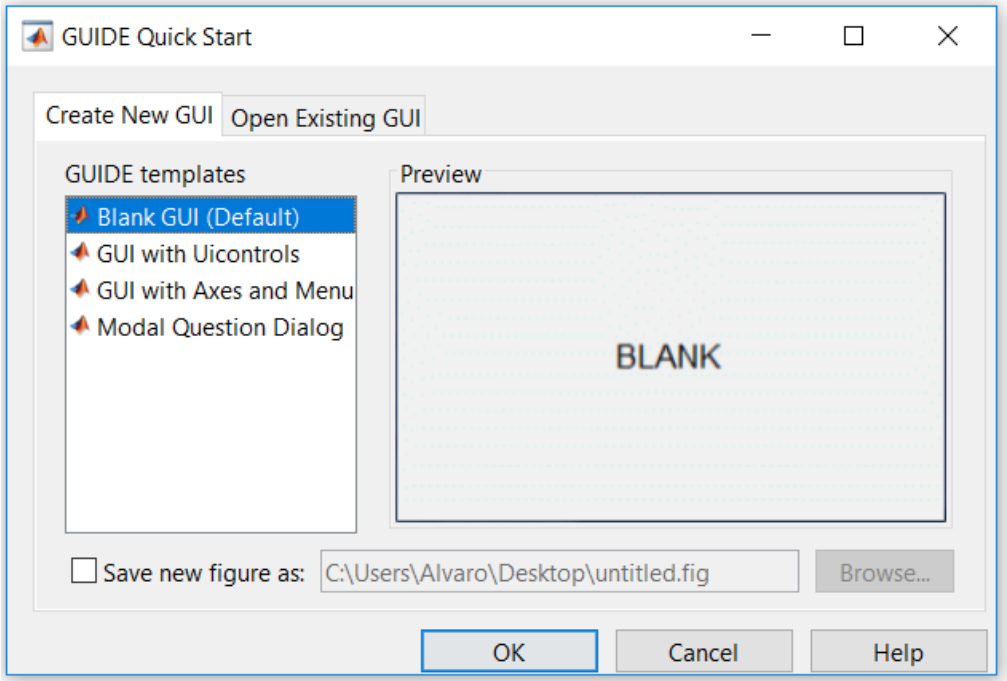

Fig. 2.3 Pantalla con plantillas de GUIDE.

Se encuentran diferentes plantillas ya configuradas con opciones básicas. Para el proyecto se utilizará la plantilla por defecto “Blank GUI” debido a que no tiene ninguna configuración hecha y será más sencillo para el diseño que queremos realizar.

Seleccionando esta opción se abre la ventana de la figura 2.4.

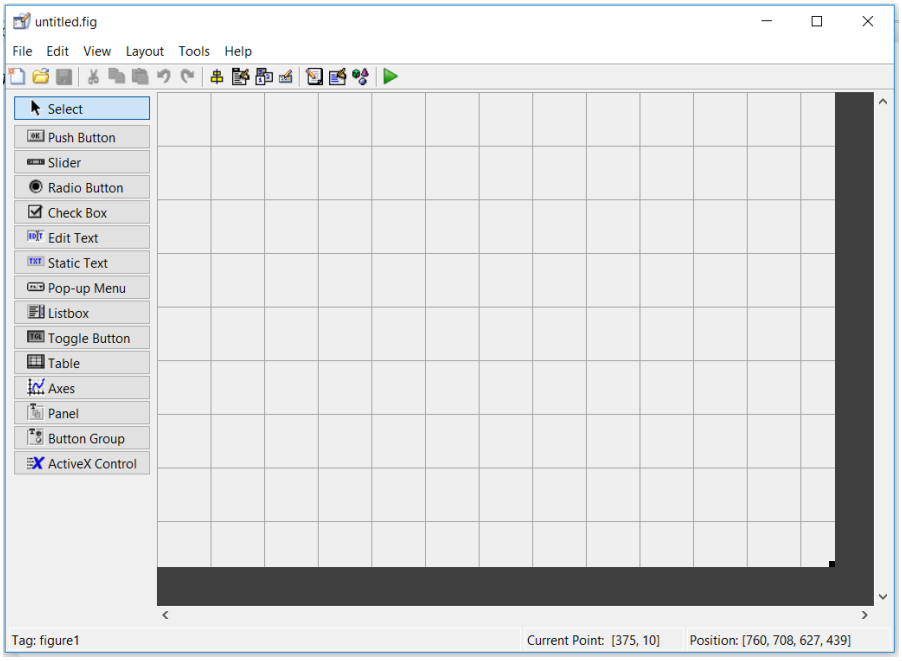

Fig. 2.4 Pantalla con plantillas de GUIDE.

En la parte izquierda de la figura se encuentra los componentes que puede contener la interfaz de usuario, en la derecha encontramos la zona de diseño donde se irán poniendo los componentes que forman la GUI y, por último, en la parte superior localizamos las opciones y menús que contiene GUIDE.

2.2.2. Property inspector

La mayoría de componentes tienen un *Callback* asociado que define que acción realizara un determinado componente cuando el usuario lo active.

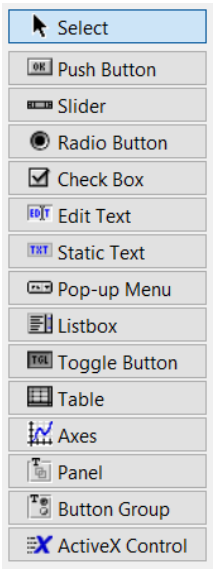

Fig. 2.5 Componentes disponibles en GUIDE.

Para editar los componentes de la GUI y ver que opciones tienen disponibles, se pulsa el botón derecho del ratón sobre un componente que se encuentre dentro del área de diseño de GUIDE.

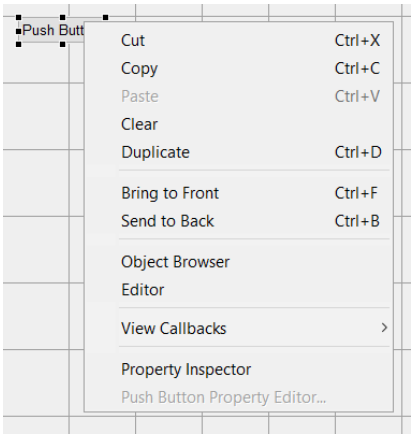

Fig. 2.6 Opciones disponibles para un botón.

Con la opción *Property Inspector* se puede personalizar cada componente que se tiene disponible en GUIDE. Dependiendo del componente seleccionado las propiedades disponibles serán diferentes.

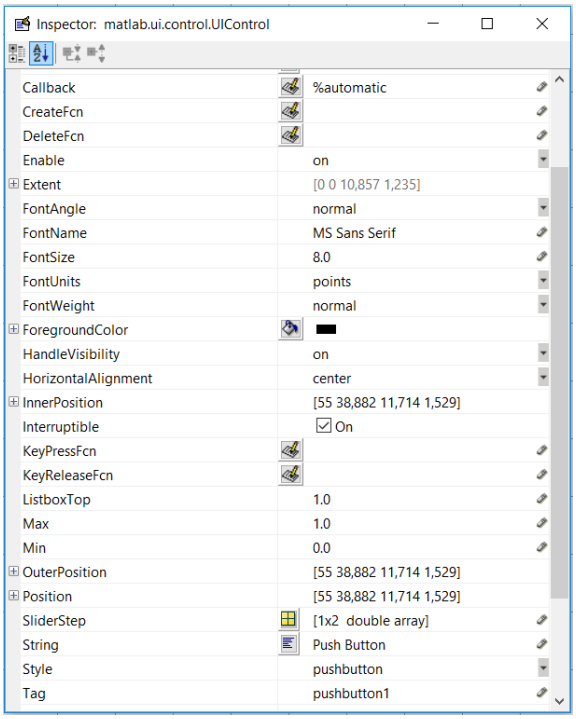

Fig. 2.7 Ventana Property Inspector.

Además de poder personalizar el tipo y tamaño de la fuente, existen dos propiedades muy importantes (*String* y *Tag*) que se deben modificar obligatoriamente si se quiere tener un mayor control sobre los componentes de la GUI.

La propiedad *String* hace referencia al texto que aparecerá sobre el componente (Caja de texto, botón, texto estático). El *Tag* es el identificador del componente, es importante para mostrar o extraer datos de un determinado componente.

A medida que se está desarrollando una GUI, se van incorporando componentes hasta el punto de ser una dificultad encontrar la función asociada a este componente. Por esta razón, una opción a tener en cuenta es *View Callbacks*, ya que muestra la parte del programa donde se encuentra la función con el código que realiza el componente seleccionado.

2.2.3. Archivos generados

Al crear una aplicación con GUIDE se generan automáticamente dos archivos: fichero .fig y fichero .m. El primer archivo con extensión .fig contiene los elementos visuales que componen la GUI, la posición de cada uno de ellos y su configuración. Por otro lado, el archivo .m contiene las funciones (*Callbacks*) de los elementos gráficos y el código que controla la interfaz gráfica.

A medida que se van incorporando nuevos elementos en la GUI, se genera automáticamente el *Callback* en el fichero .m, que se tendrá que programar para que realice la función deseada.

Una vez creados los dos archivos y programado el código se tiene que verificar el funcionamiento de la aplicación, para ello se utiliza la ventana de comandos de MATLAB escribiendo el nombre del fichero o abriendo el fichero .m con el editor y ejecutarlo mediante el botón RUN.

Es importante verificar el funcionamiento de cada componente antes de incorporar un nuevo elemento en la interfaz gráfica.

2.3. Handles y variables globales

Durante la ejecución de una GUI se llevan a cabo varias operaciones que generan datos. Estos datos, son utilizados internamente por otras funciones para garantizar el funcionamiento del programa, por lo tanto, deben ser guardados para ser usados en algún otro momento.

Para enviar y recibir datos dentro de la GUI se utiliza la estructura *handles*. Todas las funciones que forman la interfaz gráfica tienen como argumento de entrada la estructura *handles*, por ello, para utilizar una variable en otras funciones se usa la sintaxis siguiente:

```
handles.variable=variable;
```

Para guardar los datos de la estructura *handles* se utiliza la función *guidata* [18]:

```
guidata(hObject, handles);
```

Una vez guardado, para acceder a la variable desde otra función se debe utilizar la sintaxis:

```
variable=handles.variable;
```

Para asegurar que todos los cambios que se realizan durante la ejecución de la aplicación sean guardados se debe añadir la línea de código de *guidata* al final de cada función.

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

% CODIGO

guidata(hObject, handles);
```

Fig. 2.8 Ejemplo de estructura para guardar datos.

Otra forma de intercambiar datos sin tener que utilizar la estructura *handles*, es a través de variables globales [19]. Para declarar un variable global es necesario

añadir una línea de código donde se indiquen estas variables. Se utiliza la sentencia *global* seguida de las variables que queremos convertir en globales.

Una vez declaradas se asigna el valor correspondiente a cada variable. Al hacer esto ya se podrían acceder a estos valores desde otra función o GUI, siempre declarando mediante *global* las variables que se van a utilizar.

```
function pushbutton1_Callback(hObject, eventdata, handles)
global X Y

X=5;
Y=8;

guidata(hObject, handles);
```

Fig. 2.9 Ejemplo de estructura para variables globales.

Es importante seguir un orden antes de declarar variables globales, ya que si damos un valor a una variable y después la convertimos en global obtendremos un error en la ventana de comandos.

2.4. Gráficos

Una de las partes más importantes que debe tener una GUI es la representación de datos. Para ello, es necesario tener gráficos implementados en la interfaz.

Para poder presentar datos en MATLAB se utiliza la función *plot* [20], pero para que se muestren en los gráficos de nuestra GUI es necesario añadir una línea código [21] antes de esta función.

```
axes(handles.graphic);
```

Donde “graphic” hace referencia al nombre del gráfico donde se van a mostrar los resultados.

2.5. OpeningFcn

Una de las funciones más importantes que se generan automáticamente cuando creamos la GUI es el *OpeningFcn*. Esta función es la primera en ser ejecutada cuando se arranca la interfaz gráfica, por ello es muy útil para declarar variables con valores por defecto.

```
% --- Executes just before nombre_GUI is made visible.
function nombre_GUI_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for nombre_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

Fig. 2.10 Función OpeningFcn por defecto.

CAPÍTULO 3. ESTRUCTURAS DE DATOS DE ENTRADA

El software desarrollado en este TFG trabaja con datos obtenidos del sistema LIDAR multicanal de la UPC [6,14], que en ocasiones han pasado previamente por algún tipo de pre-procesado. Dependiendo de esto, los datos disponibles están en alguna de las estructuras que se presentan a continuación.

3.1. Archivos de texto con extensión .m

Estos archivos de datos están elaborados como *scripts* de MATLAB. Contienen la información de una única variable matricial que recoge las medidas brutas obtenidas del sistema LIDAR. Las variables que se definen en estos ficheros tienen los siguientes valores de interés:

- **set1a**: canal analógico de potencia total a 355 nm.
- **set8a**: canal analógico de despolarización a 355 nm.
- **set6a**: canal analógico de potencia total a 532 nm.
- **set7a**: canal analógico de despolarización a 532 nm.

Estas medidas deben experimentar un pre-procesado que incluye la sustracción de un offset y un realineamiento de muestras que corrija el adelanto en el arranque del muestreo del conversor analógico-digital [4].

En el caso de trabajar con medidas brutas en las que, previamente, se ha realizado una combinación de detección analógica y de foto-conteo para optimizar margen dinámico y relación señal a ruido [16], en todos los casos la variable es denominada **set6g**.

Cada fila de la matriz contiene un número típico de 8190 muestras correspondientes a valores de altura correlativa, separadas por una resolución típica de 3.75 m; esto implica una altura mínima de 0 m y máxima de 30708.75 m.

Las diferentes filas se corresponden con promedios realizados durante 1 minuto de medida, con un valor típico de 1200 ecos promediados durante ese minuto. El número de filas se corresponde con la duración de la medida, que puede oscilar desde 15 minutos (15 filas) típicos de las medidas correspondientes a calibraciones hasta 150 minutos (150 filas) para medidas realizadas de noche.

El programa debe trabajar con estos ficheros con independencia de la duración de las medidas y, por tanto, del número de filas de la variable.

3.2. Archivos de almacenamiento de datos de MATLAB con extensión .mat

Estos archivos contienen medidas que ya han sido pre-procesadas con un software que no es objeto del presente TFG.

En todos los casos, la variable de interés se denomina **r2p** y está formada por un vector de muestras. Este vector contiene un promediado temporal de las muestras obtenidas durante los minutos que ha durado la medida. Durante la elaboración de este promediado se han realizado ya las operaciones de alineamiento y sustracción de offset que era necesario aplicar a las medidas contenidas en ficheros .m, que ahora no será necesario repetir.

Una diferencia adicional relevante con las muestras contenidas en los ficheros .m es que, en el caso de la variable **r2p**, cada muestra ha sido multiplicada por el cuadrado de la distancia (en km) correspondiente a la misma [17]. Esto debe ser tenido en cuenta en el procesado subsiguiente.

CAPÍTULO 4. DISEÑO Y FUNCIONAMIENTO DEL PROGRAMA

En este capítulo, se detalla el funcionamiento de las diferentes GUI que componen este proyecto, así como la descripción de las diferentes opciones que se pueden encontrar para cada interfaz y sus archivos generados.

4.1. Diseño GUI

Una de las partes más importantes cuando se crea una interfaz gráfica es el diseño, ya que tiene que ser amigable e intuitivo de cara al usuario. Es de gran ayuda que todos los elementos de la interfaz (botones, cuadros de texto, etc.) estén bien ubicados y que el diseño permita futuras modificaciones. Para lograr esto, es importante entender qué tipo de archivos y variables son introducidas por el usuario, como se mostrarán los datos por pantalla y donde se guardarán los resultados, para así seguir un orden lógico para cada acción.

El primer paso que se debe realizar a la hora de hacer el diseño es dibujar en papel la apariencia que se quiere tener para la GUI. Esto supone un gran ahorro de tiempo, ya que si se hace un buen diseño que no necesite modificaciones el proceso de implementar la interfaz es mucho más rápido.

A medida que se va creando la GUI hay que ir dejando huecos para futuras incorporaciones o elementos que no se tuvieron en cuenta en un primer diseño.

Todas las interfaces graficas que se muestran en los próximos capítulos se crearon a partir de un boceto diseñado acorde a las necesidades requeridas por los programas originales.

4.2. Ficheros del programa

El proyecto contiene una serie de archivos que se utilizarán a lo largo de la ejecución del programa. Está formado por cinco interfaces gráficas, donde cada una de ellas está compuesta por un archivo .m y .fig. Por otro lado, encontramos los scripts necesarios para la representación de datos y los iconos que se muestran en el menú principal.

Todos los archivos se localizan en un mismo directorio, compuesto de diversas carpetas donde se guardarán los resultados obtenidos durante la ejecución del programa.

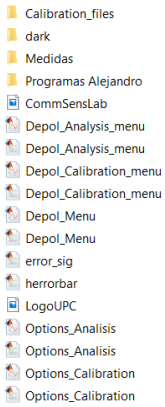

Fig. 4.1 Archivos del programa.

4.3. Menú Principal (Depol menu)

Una vez se ejecuta el programa nos encontramos con el menú principal de la interfaz gráfica, en este punto, el usuario debe elegir una de las tres opciones que se presentan (*Calibration, Analysis* o *Close*) para realizar la operación que se desee.

Por lo general, si el usuario desea trabajar con medidas nuevas y no dispone de un archivo de calibración previo se seleccionará la opción *Calibration.* Una vez obtenidos los resultados que queremos se volvería a la GUI principal para acabar seleccionado la opción *Analysis.*

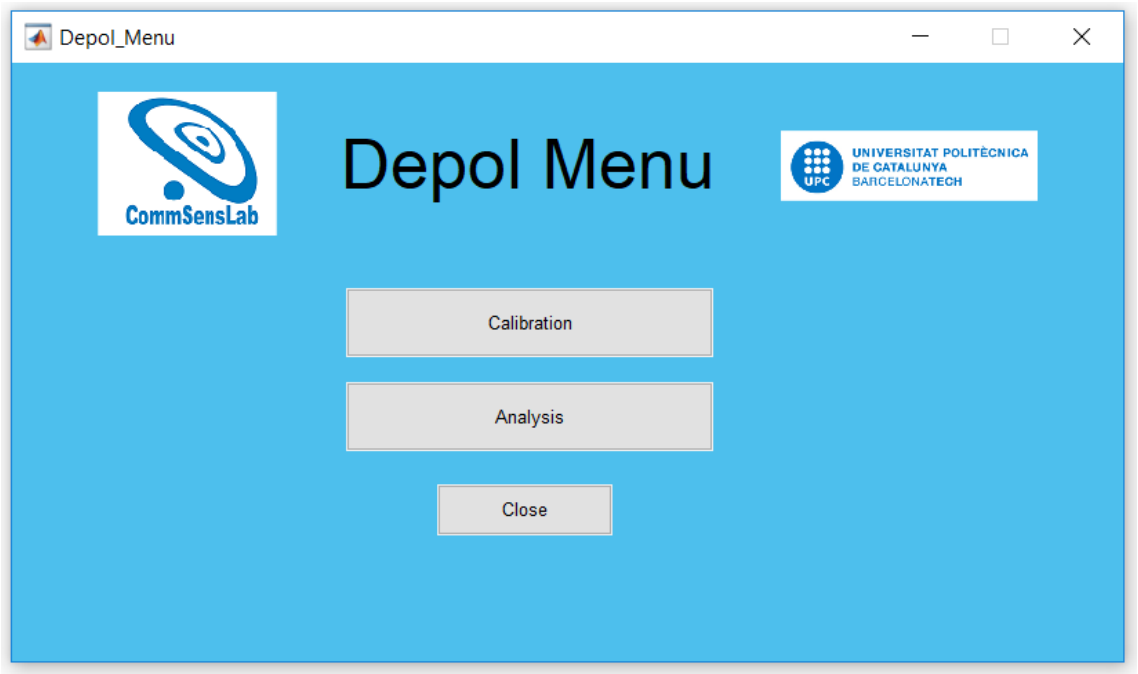

Fig. 4.2 Menú principal.

Por otro lado, si el usuario ya dispone de un archivo de calibración se elegiría la opción *Analysis* directamente, sin pasar por la opción de *Calibration*.

Finalmente, si el usuario no desea realizar más operaciones y desea cerrar la aplicación se seleccionaría la opción *Close* del menú.

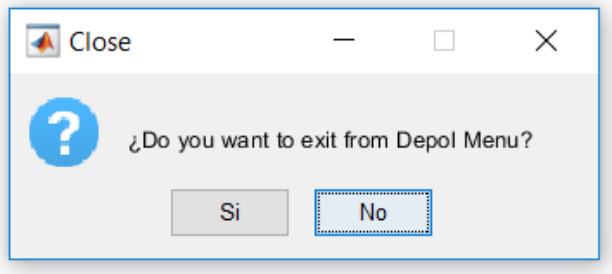

Fig. 4.3 Mensaje de salida.

4.3.1. Diagrama de flujo

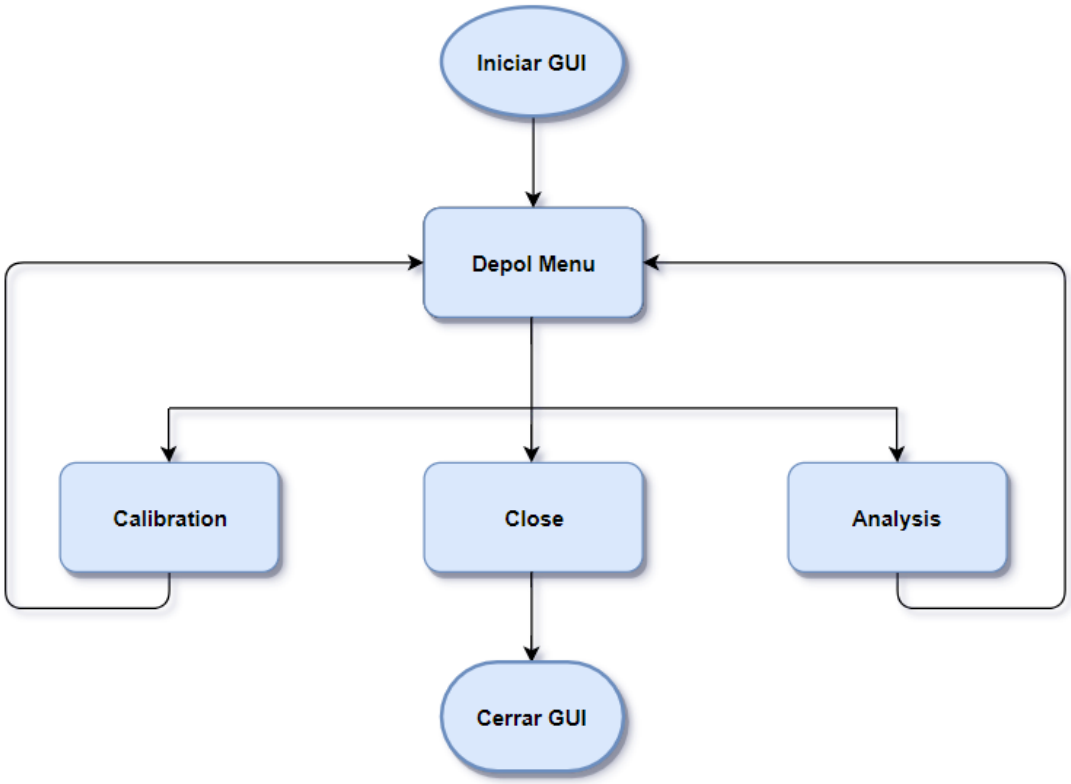

Fig. 4.4 Diagrama de flujo Depol Menu.

4.3.2. Programación Depol Menu

Como se puede observar, la interfaz principal (Fig. 4.2) contiene dos imágenes y tres botones, además de un texto estático con el nombre del menú principal.

Para mostrar las dos imágenes sobre los ejes de coordenadas se utiliza la función *imshow* [22], que muestra una imagen que se encuentre almacenada en el directorio donde se encuentran los archivos del programa (Fig. 4.1). Esta función se declara en el *OpeningFcn* ya que es la primera rutina que se ejecuta al abrir el programa.

```
% --- Executes just before Main is made visible.
function Main_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Main (see VARARGIN)

axis off

axes(handles.axes1)
imshow('LogoUPC.PNG')

axes(handles.axes2)
imshow('CommSensLab.PNG')

% Choose default command line output for Main
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

Fig. 4.5 Código OpeningFcn del Depol Menu.

Una vez seleccionada la opción con la que se va a trabajar (*Calibration, Analysis* o *Close*) el menú principal se cierra automáticamente abriendo en consecuencia la opción seleccionada. Se ha programado de esta forma para evitar errores o un cruce de variables, debido a que muchas variables tienen el mismo nombre para la opción de *Calibration* y *Analysis*.

Para abrir la GUI de calibración dentro del *Callback* del botón “Calibration”, se programa las líneas de código (Fig. 4.6) donde se llama a la GUI con el nombre Depol_Calibration_menu y al mismo tiempo se cierra el Depol Menu.

Para Análisis de Medidas, se utilizaría el mismo método cambiando únicamente el nombre de la GUI que se quiere abrir, en este caso Depol_Analysis_menu.

```
% --- Executes on button press in calibracion.
function calibracion_Callback(hObject, eventdata, handles)
% hObject    handle to calibracion (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Depol_Calibration_menu;
close(handles.main);

% --- Executes on button press in analisis_medidas.
function analisis_medidas_Callback(hObject, eventdata, handles)
% hObject    handle to analisis_medidas (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
Depol_Analysis_menu;
close(handles.main);
```

Fig. 4.6 Código para abrir GUI de Calibración.

Por último, cuando se selecciona la opción *Close* aparece una ventana con un mensaje de salida (Fig. 4.3). Este botón contiene las funciones *questdlg* [23] y *strcmp* [24].

Con *questdlg* se crea un cuadro de dialogo en forma de pregunta y con *strcmp* se comparan dos elementos, si son iguales devuelve 1 (true) de lo contrario devuelve 0 (false). Al elegir la opción “No” se obtiene como resultado *true*, por lo tanto, se vuelve al menú principal mediante el *return*. Por otro lado, si se elige “Si” el resultado de la comparación seria *false* y por consecuencia se cerraría la GUI mediante *close*.

```
% --- Executes on button press in Close_main.
function Close_main_Callback(hObject, eventdata, handles)
% hObject    handle to Close_main (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

qt=questdlg(' ¿Do you want to exit from Depol Menu?',...
'Close','Si','No','No');

if strcmp(qt,'No')
    return;
end

 close(handles.main);
```

Fig. 4.7 Código mensaje de salida.

4.4. Calibración

Una vez seleccionada la opción *Calibration* desde el menú principal, se abre la GUI de calibración (Fig. 4.8). Esta interfaz se compone de dos archivos: “Depol_Calibration_menu.fig” donde se encuentra el diseño de la GUI y “Depol_Calibration_menu.m” que contiene todos los *Callbacks* necesarios para poder obtener y representar datos durante el funcionamiento del programa. Por lo tanto, se puede considerar este archivo del proyecto como uno de los principales (junto al archivo de análisis de medidas que se explicará más adelante en el apartado 4.5).

La GUI está formada por once botones, que se irán explicando en los próximos apartados, cuatro textos estáticos y dos cuadros de texto editables. Además, están integradas cuatro graficas donde se irán mostrando los resultados obtenidos al usuario.

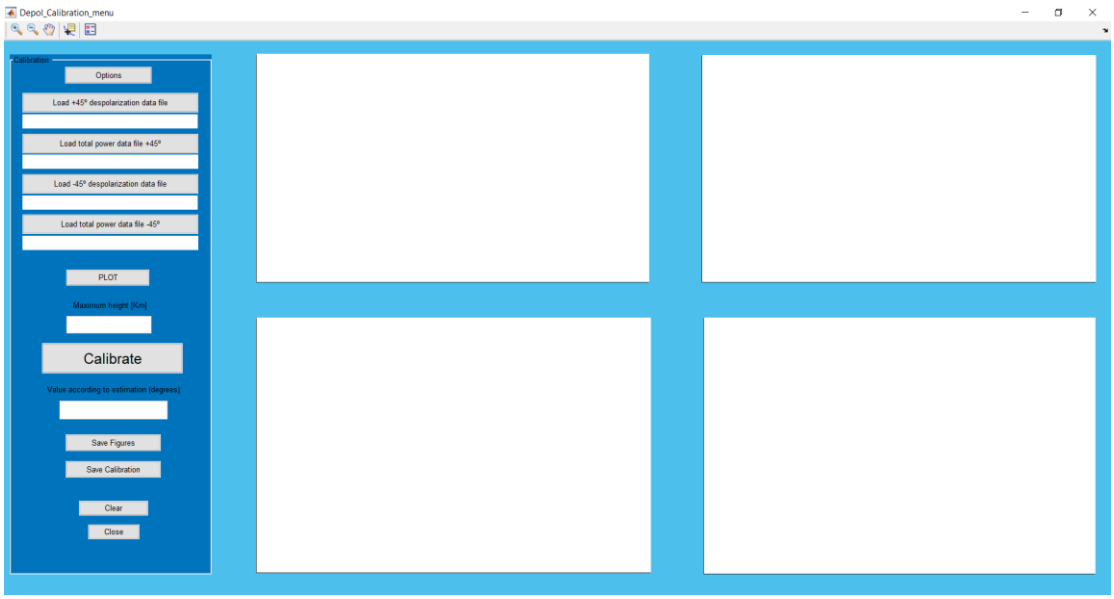

Fig. 4.8 GUI de Calibración.

4.4.1. Diagrama de flujo

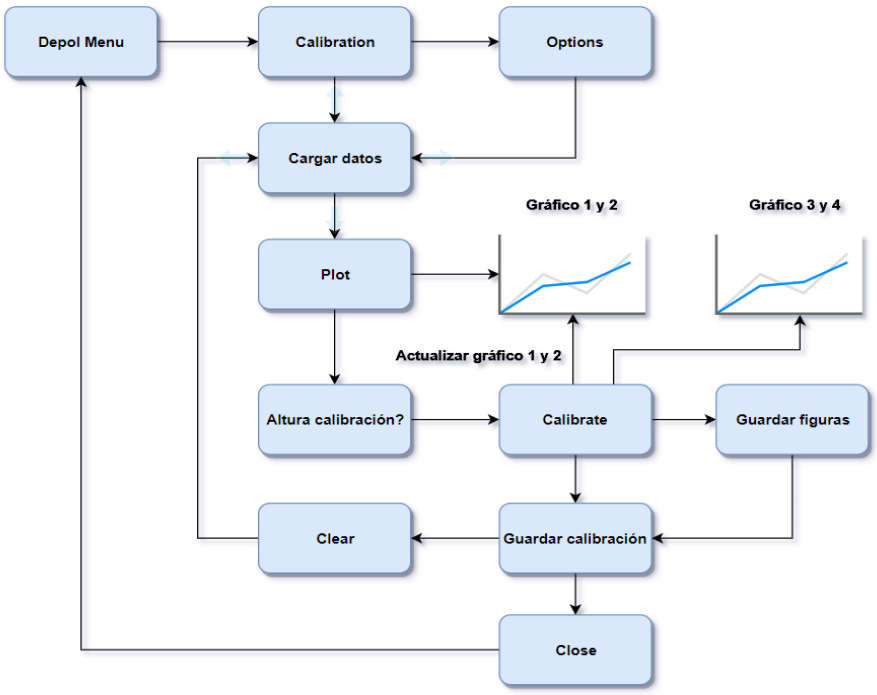

Fig. 4.9 Diagrama de flujo *Calibration.*

4.4.2. Variables iniciales Calibración (OpeningFcn)

Una de las partes más importantes de la GUI de calibración es el *OpeningFcn*, ya que contiene todas las variables por defecto que permiten que el programa funcione correctamente. A continuación, se explicará las diferentes variables que se encuentran dentro de esta función y cómo funcionan internamente en la interfaz.

Lo primero que se programa en esta función es guardar el directorio donde se encuentra la GUI, seguidamente se añade este directorio al *path* de Matlab [25].

```
% --- Executes just before Depol_Calibration_menu is made visible.
function Depol_Calibration_menu_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Depol_Calibration_menu
handles.output = hObject;

W=what;
handles.W=W.path;
addpath(handles.W);
```

Fig. 4.10 Directorio en OpeningFcn.

Una vez especificado el directorio, se declaran algunas de las variables más importantes que contiene esta GUI. A estas variables se les asigna un valor por defecto que se puede modificar a través del botón de opciones de la interfaz. Además, con la ayuda de la función *assignin* [26] cargamos el valor de estas variables en el workspace de MATLAB.

Para facilitar el intercambio de datos entre las diferentes funciones se ha optado por declarar estas variables como tipo global (ver apartado 2.3).

```
global Nofilt Dmax deltaR bin_shift_a bin_shift_ac deltaM532 deltaM355 dire S A

% Iniciamos variables globales con valores por defecto
Nofilt=133; %samples
assignin('base','Nofilt',Nofilt)
Dmax=10; %Km
assignin('base','Dmax',Dmax)
deltaR=3.75e-3; %Km
assignin('base','deltaR',deltaR)
bin_shift_a=25; %samples
assignin('base','bin_shift_a',bin_shift_a)
bin_shift_ac=28; %samples
assignin('base','bin_shift_ac',bin_shift_ac)
deltaM532=3.8e-3;
assignin('base','deltaM532',deltaM532)
deltaM355=8e-3;
assignin('base','deltaM355',deltaM355)
```

Fig. 4.11 Variables globales OpeningFcn.

Otra variable global muy importante es aquella que contiene el directorio de la GUI (*dire*) y las utilizadas para seleccionar el tipo de archivo (explicación en el apartado 4.4.3).

```
% Asignamos directorio a variable global
dire=handles.W;
assignin('base','dire',dire)

% Variables de selección de archivo
A='m';
S=1;
```

Fig. 4.12 Variable global de directorio y selección de archivos.

Por último, a modo estético, se quitan los valores por defecto cargados en los ejes X e Y de las cuatro graficas que están integradas en la GUI.

```
% Quitamos valores por defecto de los ejes
set(handles.axes1,'xtick',[], 'ytick',[])
set(handles.axes2,'xtick',[], 'ytick',[])
set(handles.axes3,'xtick',[], 'ytick',[])
set(handles.axes4,'xtick',[], 'ytick',[])
```

Fig. 4.13 Código para quitar valores ejes.

A continuación, se describirá el código ejecutado al presionar cada uno de los botones de la interfaz.

4.4.3. Opciones

El botón “Options” abre una ventana con las opciones disponibles para la GUI de calibración. El código programado en el *Callback* del botón permite abrir la interfaz de opciones paralelamente a la GUI de calibración.

```
function options_Callback(hObject, eventdata, handles)

Options_Calibration;

uiwait

guidata(hObject, handles);
```

Fig. 4.14 Código para abrir ventana de opciones.

Cuando se abre la ventana de opciones (Fig. 4.15) el usuario puede modificar los valores cargados por defecto. Además, nos encontramos con un pop-up desplegable donde se puede elegir con qué tipo de archivos se va a trabajar (archivos .m o .mat).

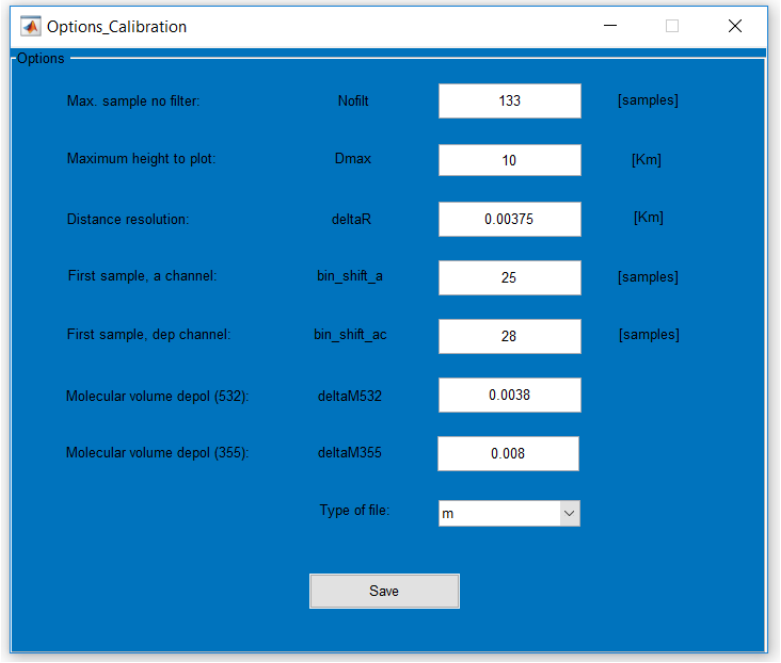

Fig. 4.15 Opciones para calibración.

Los valores de estas variables se definen en el archivo ‘‘Depol_Calibration_menu.m’’, más concretamente en la función *OpeningFcn* (Fig. 4.11)

Una vez establecido el valor de cada variable, desde el OpeningFcn del archivo “Options_Calibration.m” se ejecuta el código para mostrar los valores en los cuadros de texto de la figura 4.15. Para ello, se utiliza el comando *set* [27] con el cual se asignan un valor a un determinado componente, en este caso, un cuadro de texto. Cada cuadro de texto tiene un identificador único (Tag) para poder asignar u obtener un valor.

```
global Nofilt Dmax deltaR bin_shift_a bin_shift_ac deltaM532 deltaM355

set(handles.nofilttext,'String',Nofilt) %samples

set(handles.dmaxtext,'String',Dmax) %Km

set(handles.deltartext,'String',deltaR) %Km

set(handles.bin_shift_a_text,'String',bin_shift_a)

set(handles.bin_shift_ac_text,'String',bin_shift_ac)

set(handles.deltam532text,'String',deltaM532)

set(handles.deltam355text,'String',deltaM355)
```

Fig. 4.16 OpeningFcn en el botón de opciones.

Para seleccionar el tipo de archivo se ha programado un pop-up menú desplegable, donde el usuario puede elegir qué tipo de archivo se va a utilizar.

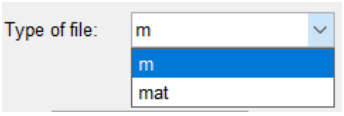

Fig. 4.17 Selección de tipo de archivo.

Por defecto, como se puede ver en la figura 4.17 el tipo de archivo es m. Para declarar estas condiciones iniciales se utilizan las variables globales *A* y *S* (Fig. 4.12). La variable *A* indica el tipo de archivo (m o mat) y la variable *S* indica la posición que ocupa la variable *A* en el desplegable, es decir, si ocupa la posición 1 (arriba) o posición 2 (abajo). Dependiendo de la elección la variable *A* se actualizará al tipo de archivo seleccionado.

```
% --- Executes on selection change in selec_files.
function selec_files_Callback(hObject, eventdata, handles)

global A S
contents = cellstr(get(hObject,'String'));
popChoice = contents(get(hObject,'Value'));

if (strcmp(popChoice,'m'))
    A='m';
    S=1;
elseif (strcmp(popChoice,'mat'))
    A='mat';
    S=2;
end
```

Fig. 4.18 Código del Pop-up menú.

Una vez realizada la selección se compara el contenido del pop-up menú con la función, comentada anteriormente, *strcmp* [24]. Dependiendo del resultado de la comparación las variables se actualizarán para ser utilizadas por la GUI de calibración.

Finalmente, con el botón “Save” se detecta si se ha producido cualquier modificación en las variables, para ello se utiliza la función *get* [28] con la que se obtiene qué valor tiene actualmente el cuadro de texto.

```
% --- Executes on button press in save.
function save_Callback(hObject, eventdata, handles)

global Nofilt Dmax deltaR bin_shift_a bin_shift_ac deltaM532 deltaM355

Nofilt=str2double(get(handles.nofilttext,'String'));
Dmax=str2double(get(handles.dmaxtext,'String'));
deltaR=str2double(get(handles.deltartext,'String'));
bin_shift_a=str2double(get(handles.bin_shift_a_text,'String'));
bin_shift_ac=str2double(get(handles.bin_shift_ac_text,'String'));
deltaM532=str2double(get(handles.deltam532text,'String'));
deltaM355=str2double(get(handles.deltam355text,'String'));

close Options
```

Fig. 4.19 Código del botón Save.

Si se modifica cualquier variable, el valor se actualizará automáticamente debido a que es una variable global. Por lo tanto, para cálculos posteriores se utilizará el valor más actualizado de la variable.

4.4.4. Botones de selección de datos de calibración

Para poder empezar a trabajar con la aplicación es necesario cargar una serie de archivos que contienen los datos. Nos encontramos con cuatro botones debido a que hay que cargar cuatro archivos en la GUI, dos archivos de despolarización (+ 45º y - 45º) y dos archivos de potencia total (+ 45º y – 45º).

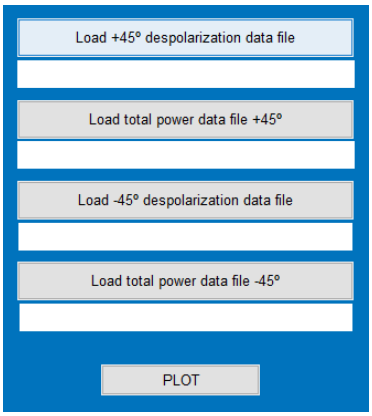

Fig. 4.20 Botones de selección de archivos.

Para cada uno de los cuatro botones nos encontramos con dos tipos de datos (ficheros en .m o .mat), dependiendo de la elección hecha en el botón de “Options” (Fig. 4.17) los datos se procesan de una forma u otra. Como hemos visto anteriormente, la variable global *A* indica el tipo de archivo seleccionado.

Para procesar correctamente los datos, en función del tipo de archivo, se utiliza un *switch* [29], donde si la variable *A* tiene como valor igual a *m* se ejecutará un código y por el contrario si es igual a *mat* se ejecutará otro código distinto.

```
global A

switch A
        case 'm'
        %Código para procesar archivos .m

        case 'mat'
        %Código para procesar archivos .mat
end
```

Fig. 4.21 Estructura procesado de datos.

Si se decide trabajar con archivos .m y se pulsa uno de los botones nos aparece una ventana para seleccionar el archivo correspondiente, por defecto la ruta donde se abre esta ventana es donde se encuentran los archivos que forman la interfaz, por ello hay que navegar entre las diferentes carpetas hasta llegar a la ruta donde se encuentran los archivos que se quieren procesar.

Un ejemplo de los archivos que pueden contener la carpeta lo podemos encontrar en la figura 4.22.

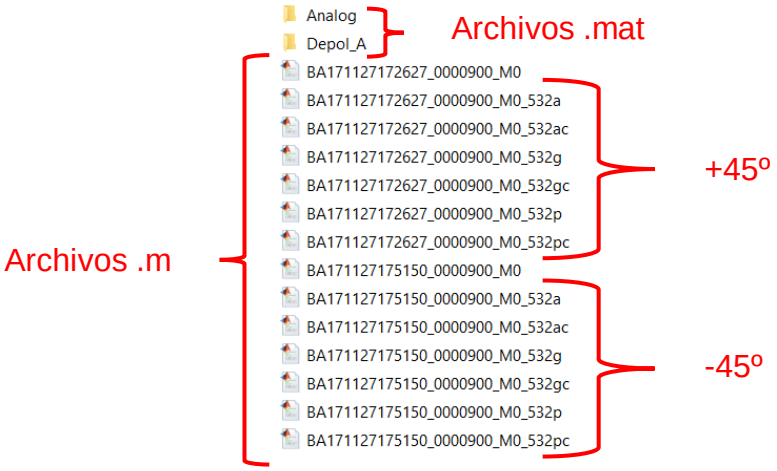

Fig. 4.22 Estructura de archivos.

Por un lado, se encuentran los archivos +45º y por otro los de -45º. Dentro de estas categorías se encuentran tres tipos de datos (a, g, p) y a la vez si son archivos de despolarización o potencia total. La forma de diferenciarlos la encontramos en el último dígito del nombre del archivo, ya que los que tienen dos letras (ac, gc, pc) hacen referencia a los archivos de despolarización, en cambio los que contienen solamente una letra (a, g, p) se refieren a los archivos de potencia total. Otro dato especificado en los archivos es la longitud de onda.

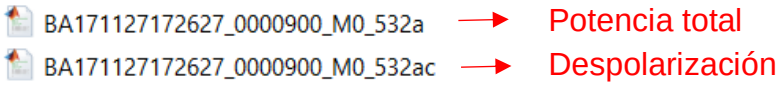

Fig. 4.23 Archivos .m tipo a.

Por otra parte, si el usuario decide trabajar con archivos .mat la estructura de selección de archivos es diferente, esto es debido a que los archivos de despolarización los encontraremos en una carpeta llamada “Depol_A” y los archivos de potencia total en la carpeta “Analog” (Fig. 4.22).

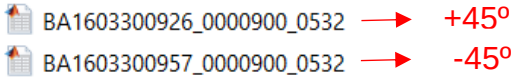

Fig. 4.24 Archivos en carpeta Depol_A.

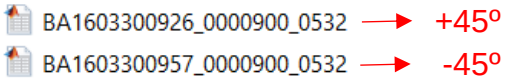

Fig. 4.25 Archivos en carpeta Analog.

El nombre de este tipo de archivo, a diferencia de los archivos .m, no contiene el tipo de dato (a, g, p), sino la longitud de onda del archivo. El tipo de dato viene definido en el nombre de la carpeta “Depol_A”, donde A hace referencia al tipo de dato.

Para la selección de los diferentes archivos se ha utilizado la función *uigetfile* de Matlab [30]. El nombre de los archivos elegidos se puede observar debajo de cada botón gracias a los textos estáticos que hay disponibles.

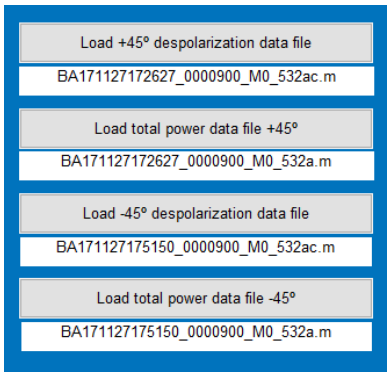

Fig. 4.26 Archivos seleccionados.

Es importante destacar que no hay un orden definido para la elección de archivos, se puede empezar a seleccionar un archivo con cualquiera de los cuatros botones, siempre que corresponda con su tipo de datos.

Una vez se pulsa un botón y se elige el archivo, el siguiente botón abrirá la ventana de selección en el último directorio que se ha abierto, esto es muy cómodo debido a que no hay que estar navegando entre carpetas de medidas, ya que puede suponer una gran dificultad encontrar la misma medida si se tienen muchas. Todo esto se ha programado con una variable de tipo global llamada *dire*, donde inicialmente contiene el directorio donde se encuentra la GUI (Fig. 4.12). Código detallado en el ANEXO 2.

4.4.5. Botón Plot

El siguiente paso después de haber seleccionado los archivos es mostrar los primeros resultados por pantalla. Si se pulsa el botón ‘‘PLOT’’ se representan los resultados sobre las dos gráficas superiores (ver Fig. 4.8).

Inicialmente la representación de datos de potencia total sobre la gráfica no era correcta, debido a que en el eje Y se mostraban valores negativos que no se querían observar, para solucionar esto se ha utilizado la función *ylim* [31].

```
% --- Executes on button press in plot.
function plot_Callback(hObject, eventdata, handles)

global Dmax deltaR r2plus r2pminus r2totalp r2totalm

Nplot=floor(Dmax/deltaR);

r=deltaR*[1:8189];

% ----------PLOT DEPOL----------
axes(handles.axes1);
plot(r(1:Nplot),r2plus(1:Nplot),r(1:Nplot),r2pminus(1:Nplot))
xlabel('Height AGL (km)')
ylabel('R^2P (mV·km^2)')
lgd=legend('Depolarization signal +45°','Depolarization signal -45°','Location','Best');
title(lgd,'Depolarization Signal')

% ----------PLOT POWER----------
axes(handles.axes2);
plot(r(1:Nplot),r2totalp(1:Nplot),r(1:Nplot),r2totalm(1:Nplot))
ylim([0 inf])
xlabel('Height AGL (km)')
ylabel('R^2P (mV·km^2)')
lgd=legend('Total received signal +45°','Total received signal -45°','Location','Best');
title(lgd,'Total Power');

guidata(hObject, handles);
```

Fig. 4.27 Código para representar los datos.

En la figura 4.28, se muestra la señal de despolarización, captada por el telescopio auxiliar, donde se pueden apreciar dos tipos de datos. En color azul se representa la señal de despolarización para + 45º y en color rojo la señal de despolarización para - 45º.

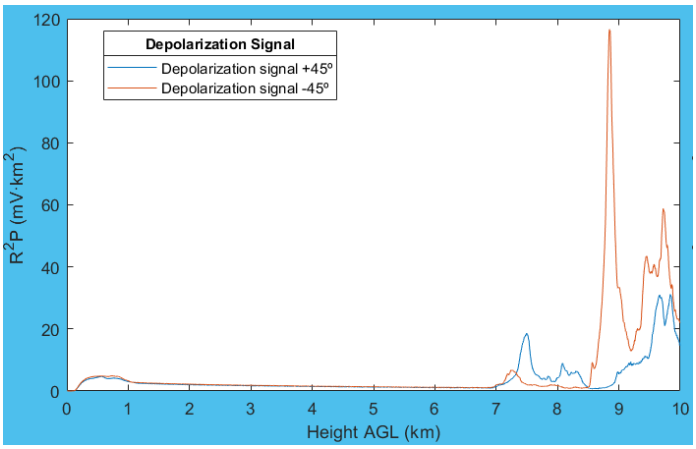

Fig. 4.28 Gráfica señal de despolarización.

En la figura 4.29 se muestra la señal de potencia total, captada por el telescopio principal, donde en color azul se representa la señal recibida para +45º y en color rojo la señal recibida para - 45º, multiplicadas por el cuadrado de la altura a fin de mostrar las características de la señal eco [17].

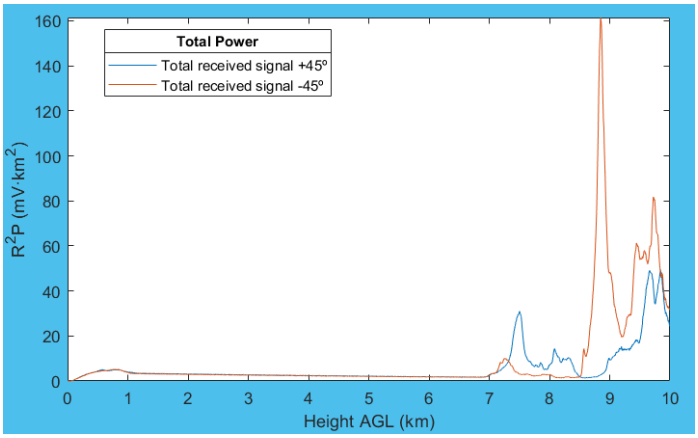

Fig. 4.29 Gráfica señal de potencia total.

Como se puede observar en los dos gráficos, los ejes X (Height AGL (km)) tienen la misma distancia (10 km), esto es debido a la variable global *Dmax* definida en apartados anteriores.

4.4.6. Calibración

En este apartado, se encuentra el proceso más importante de la GUI de calibración ya que permite obtener los parámetros $V^*(R)$ y φ_0 vistos en el apartado 1.2.

Antes de empezar la calibración es necesario introducir la distancia máxima, para ello, el usuario debe interpretar los resultados obtenidos en el apartado anterior

e introducir el valor deseado. Una vez se ha introducido el valor se pulsa el botón “Calibrate” para mostrar los resultados que faltan sobre la interfaz.

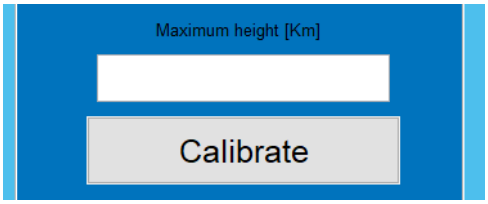

Fig. 4.30 Cuadro de texto para introducir las distancia máxima.

En los gráficos inferiores, se muestran los resultados obtenidos tras realizar esta operación de calibración.

Observando los resultados obtenidos anteriormente y para mostrar los resultados durante el proceso de calibración se ha utilizado una distancia máxima de 6 km.

En la figura 4.31 se muestra los resultados obtenidos de la ecuación **1.13**, en color rojo se representa $\delta^*(\varphi_-, R)$ y en azul $\delta^*(\varphi_+, R)$. Finalmente, en color rosa encontramos la suma de estos dos perfiles $V^*(R)$ (ecuación **1.14**).

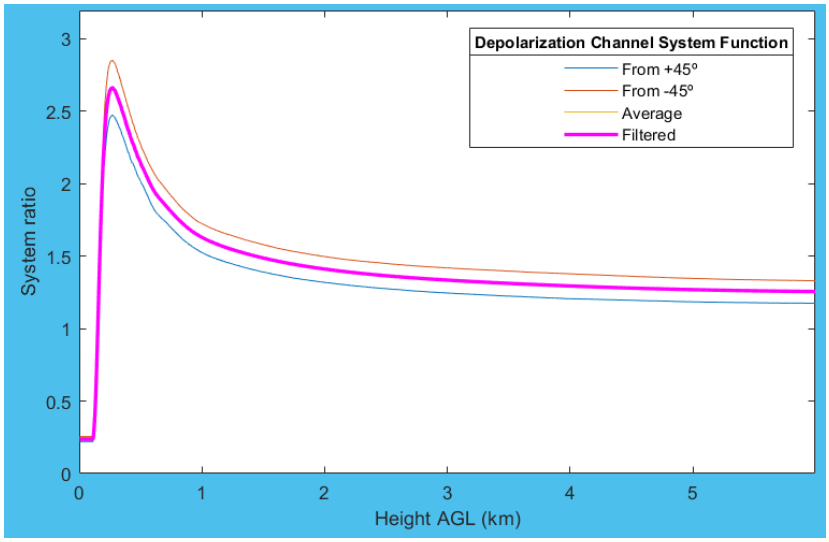

Fig. 4.31 Gráfica sistema del canal de despolarización.

Para la posición actual del polarizador se tiene que calcular el ángulo en función de la distancia máxima escogida. Si se tiene en cuenta todo el rango obtendríamos un valor del ángulo que no se aproxima al valor real y, en consecuencia, para observar un valor más cercano al real, se ha representado la gráfica en el último medio kilómetro del valor escogido para la distancia máxima.

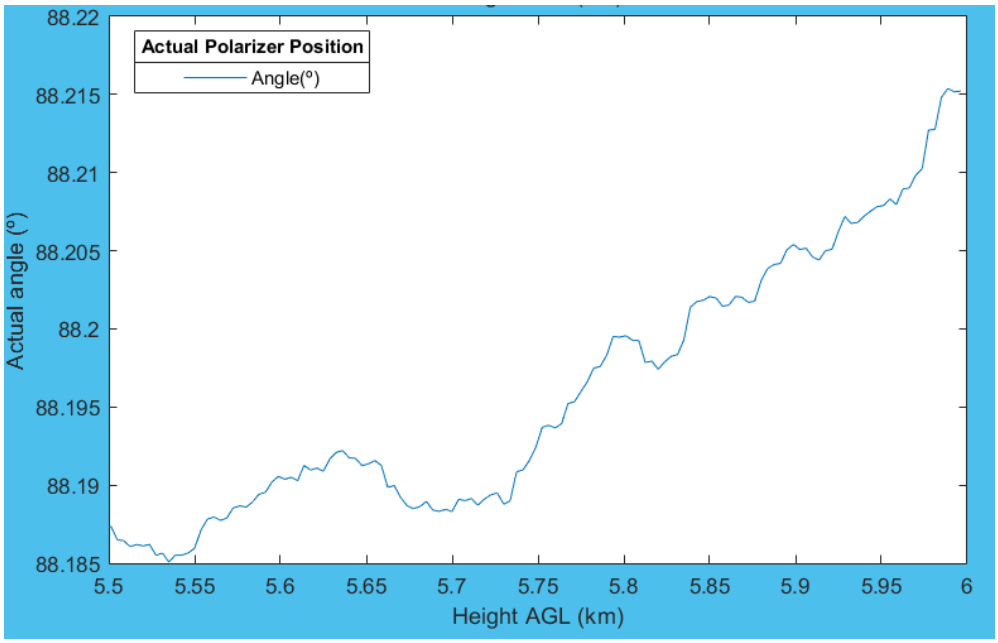

Fig. 4.32 Gráfica ángulo actual polarizador.

Por otro lado, los gráficos de señal de despolarización y potencia total se actualizan en función de la distancia máxima escogida, mostrando los datos comprendidos en el rango seleccionado:

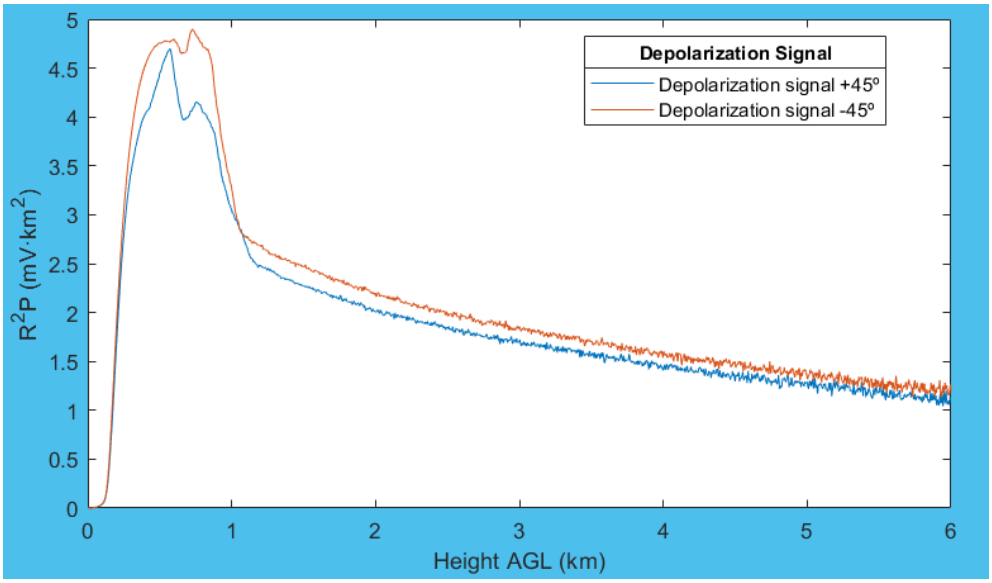

Fig. 4.33 Gráfica señal de despolarización actualizada.

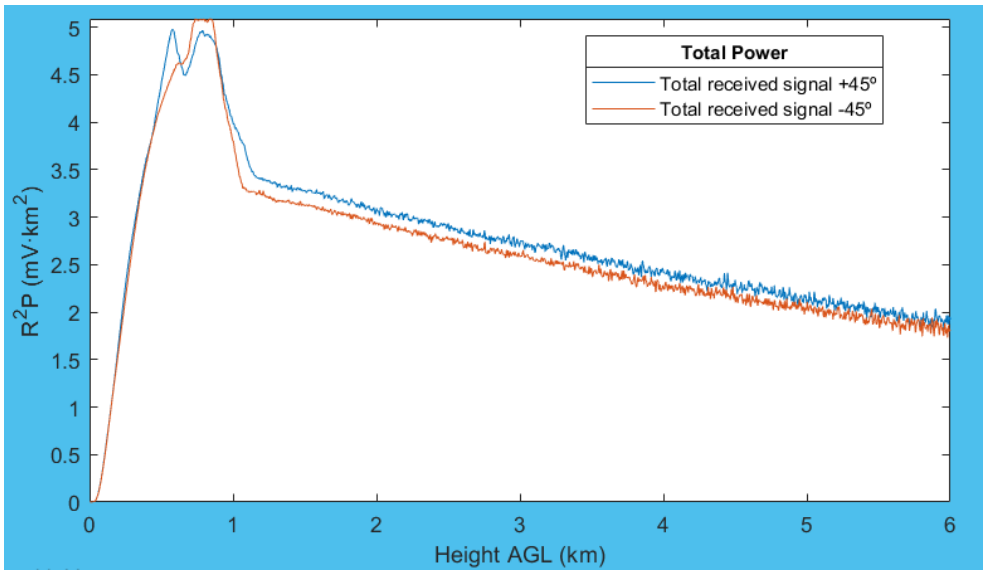

Fig. 4.34 Gráfica señal de potencia total actualizada.

Finalmente, se calcula un promedio del ángulo que se observa en la figura 4.32, este valor se muestra sobre un cuadro de texto donde el usuario puede modificar o no el contenido dependiendo de su criterio.

```
interval=find(((rmaxi-0.5)<r) & (r<rmaxi));
angulo_promedio=mean(phi0(interval));
assignin('base','angulo_promedio',angulo_promedio)
set(handles.angulo,'String',angulo_promedio);
```

Fig. 4.35 Código para calcular el promedio del ángulo.

El valor obtenido corresponde al ángulo φ_0 visto en la teoría (apartado 1.2) donde el valor ideal corresponde a 90º, por esta razón los resultados obtenidos tienen que estar entorno a los 90º ideales.

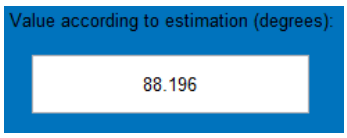

Fig. 4.36 Ángulo promedio obtenido.

4.4.7. Guardar gráficos obtenidos

Para guardar los resultados obtenidos en las gráficas se utiliza el botón “Save Figures”. Una vez pulsado el botón se le pide al usuario, con la ayuda de la función *uigetdir* [32], que elija la carpeta donde se guardarán las figuras.

Una vez elegida la carpeta y con la funcion *subplot* [33], se genera una figura con los gráficos obtenidos durante la ejecución de la interfaz, donde los ejes estan invertidos. Ademas, se ha suprimido la grafica de la figura 4.32, pero se ha incluido el valor del promedio del angulo.

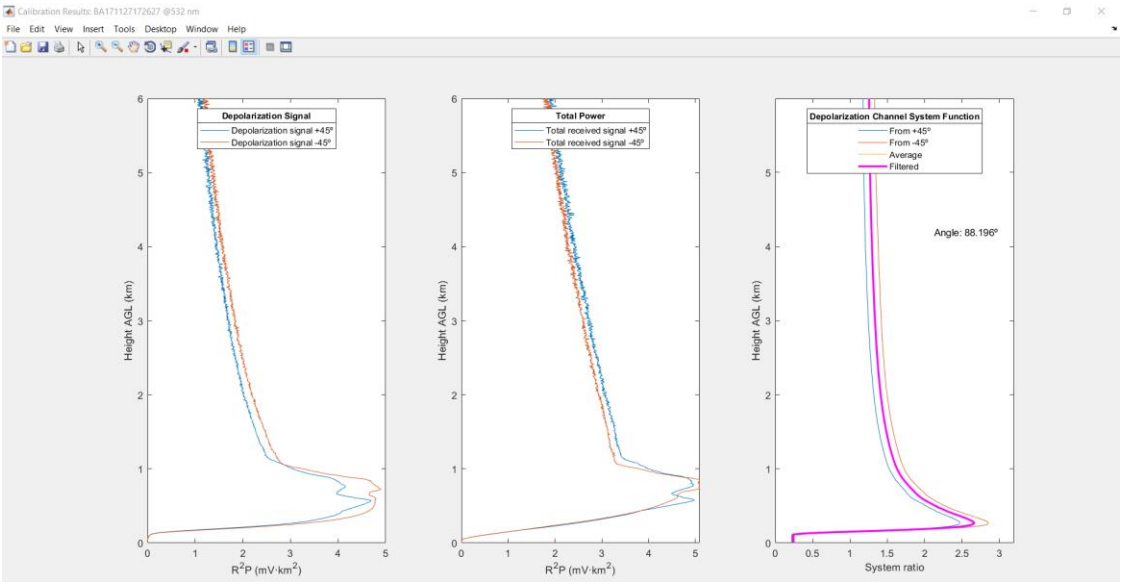

Fig. 4.37 Figura generada con subplot.

```
savedir = uigetdir(dire,'Choose folder to save the figures');
switch A

    case 'm'
        cd(savedir)
        saveas(h1,[nombre,'_',wavel,'_',  discrim, '_calibration_results'],'fig')
        saveas(h1,[nombre,'_',wavel,'_',  discrim, '_calibration_results'],'jpg')
        saveas(h1,[nombre,'_',wavel,'_',  discrim, '_calibration_results'],'tif')

    case 'mat'
        cd(savedir)
        saveas(h1,[nombre_mat,'_',wave_mat,'_',  discrim_mat, '_calibration_results'],'fig')
        saveas(h1,[nombre_mat,'_',wave_mat,'_',  discrim_mat, '_calibration_results'],'jpg')
        saveas(h1,[nombre_mat,'_',wave_mat,'_',  discrim_mat, '_calibration_results'],'tif')
end
```

Fig. 4.38 Código para guardar las figuras.

Como se puede observar en la figura 4.38, las figuras se guardan en tres formatos diferentes (fig, jpg, tif). El nombre con el que se guardan las figuras está formado por el nombre de la medida (BA171127), la longitud de onda (532) y el tipo de dato (a). Toda esta información se obtiene durante la selección de archivos (apartado 4.4.4).

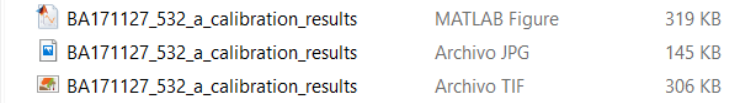

Fig. 4.39 Archivos generados al guardar figuras.

4.4.8. Guardar resultados de la calibración

Para finalizar el proceso de calibración es necesario guardar el archivo con los parámetros obtenidos y que serán utilizados en el proceso de análisis de medidas.

```
savedir = uigetdir(dire,'Choose folder to save Calibration');
if isequal(savedir,0)
     cd(handles.W)
else
switch A
case 'm'
    sysfile=['Depol_cte_Comeron', nombre, '_',wavel,'_', discrim];
    cd(savedir)
    eval(['save ', sysfile, ' System_constantf phi0def ctep_f ctem_f']);

case 'mat'

    sysfile=['Depol_cte_Comeron', nombre_mat, '_',wave_mat,'_', discrim_mat];
    cd(savedir)
    eval(['save ', sysfile, ' System_constantf phi0def ctep_f ctem_f']);
end
end
```

Fig. 4.40 Código para guardar archivo de calibración.

El archivo generado se guarda en formato .mat y el nombre con el que se guarda el archivo de calibración se obtiene durante la selección de archivos (apartado 4.4.4).

Fig. 4.41 Archivo de calibración.

Si se analiza el contenido del archivo, se pueden observar cuatro variables que se han obtenido con los cálculos internos del programa. La variable *System_constantf* corresponde a $V^*(R)$, *ctep_f* al observable para +45º ($\delta^*(\varphi_+, R)$), *ctem_f* al observable para -45º ($\delta^*(\varphi_-, R)$) y *phi0def* a φ_0.

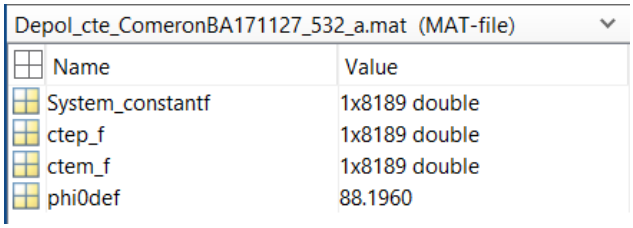

Fig. 4.42 Contenido archivo calibración.

4.4.9. Botones Clear y Close

Por último, si se desea hacer más operaciones de calibración se pulsaría el botón de “Clear”, con el cual se borraría el contenido de los gráficos de la interfaz gráfica, además de los archivos cargados y valores obtenidos en los cálculos.

Por otro lado, si el usuario ya no desea continuar con las operaciones de calibración, presionaría el botón “Close” con el que cerraría la interfaz de calibración volviendo al menú principal del programa (Depol Menu).

4.4.10. Ejemplo calibración reciente

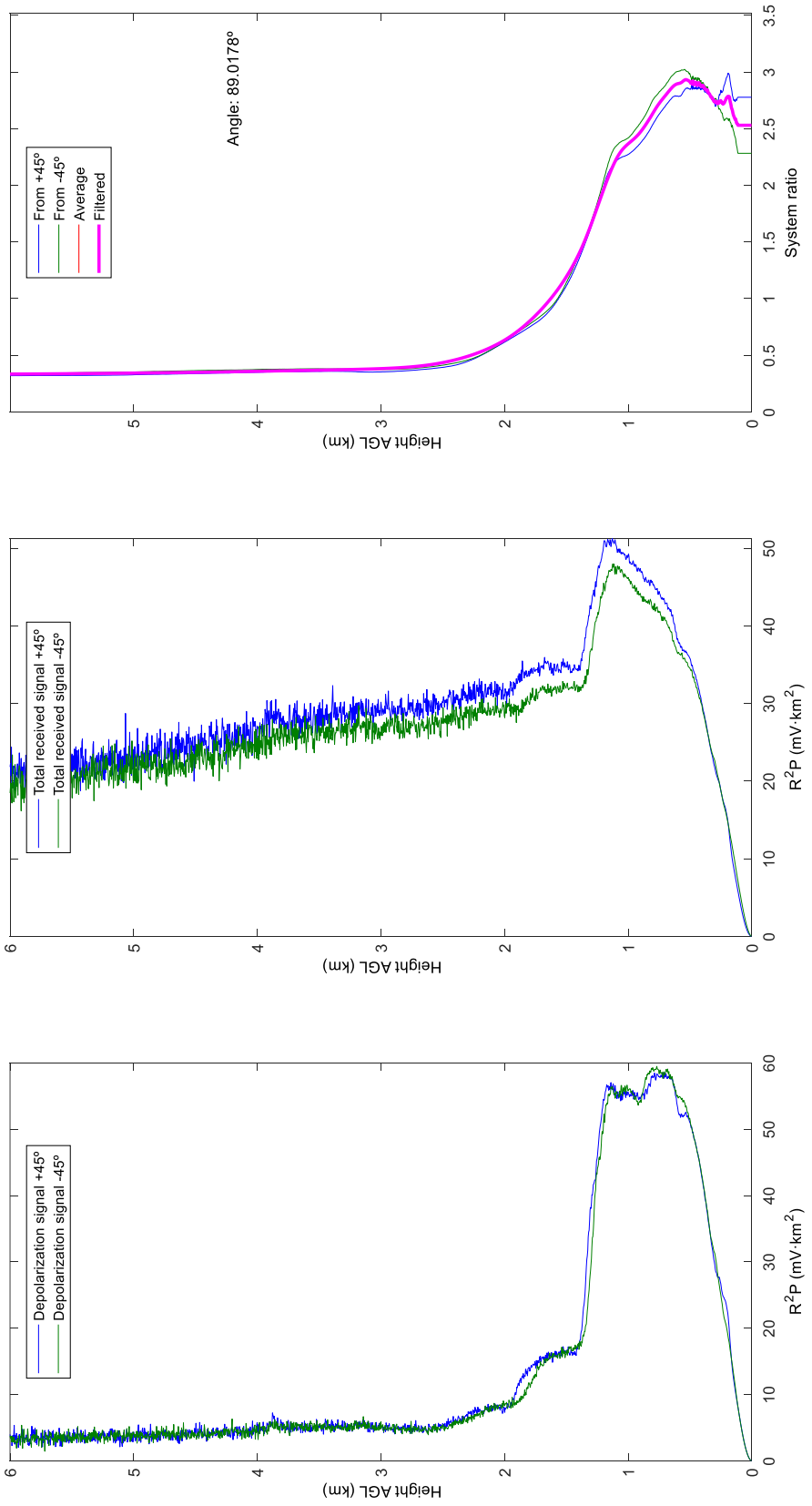

Fig. 4.43 Calibración 2 de octubre de 2018 para 532 nm.

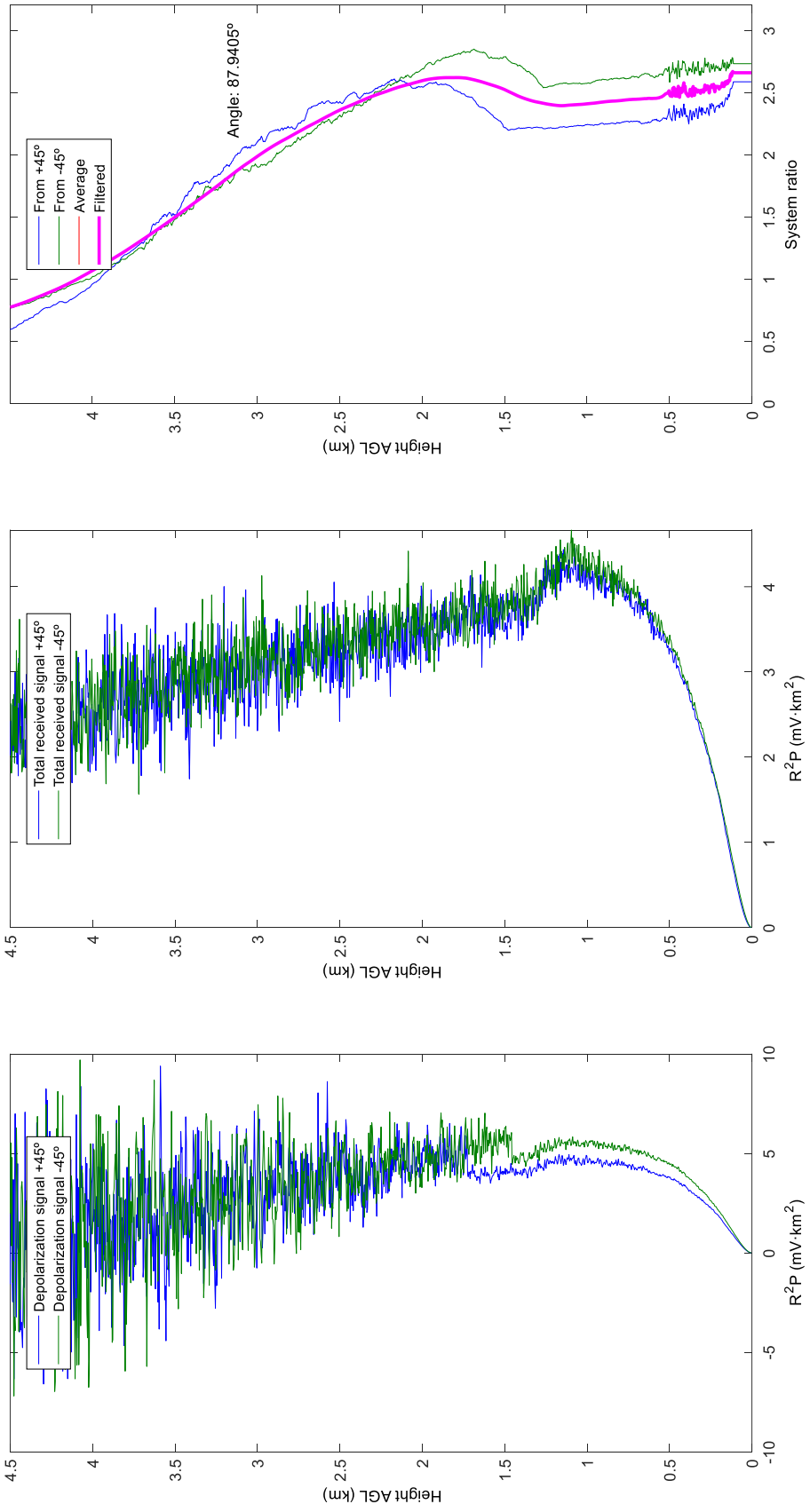

Fig. 4.44 Calibración 2 de octubre de 2018 para 355 nm.

4.5. Análisis de medidas

Una vez se obtiene el archivo de calibración se puede proceder a realizar el análisis de medidas, que permitirá calcular el observable $(\delta^*(\varphi, R))$, la despolarización de volumen $\delta^V(R)$ y de partícula $\delta^p(R)$. En el menú principal (Depol Menu) se pulsa la opción *Analysis* para abrir la GUI de análisis de medidas (ver Fig. 4.45).

La GUI está formada por dos archivos: "Depol_Analysis_menu.m" y "Depol_Analysis_menu.fig". Contiene tres gráficas para mostrar resultados, catorce botones con diferentes funciones, cinco cuadros de texto estático para mostrar información y dos cuadros de texto editables para introducir valores.

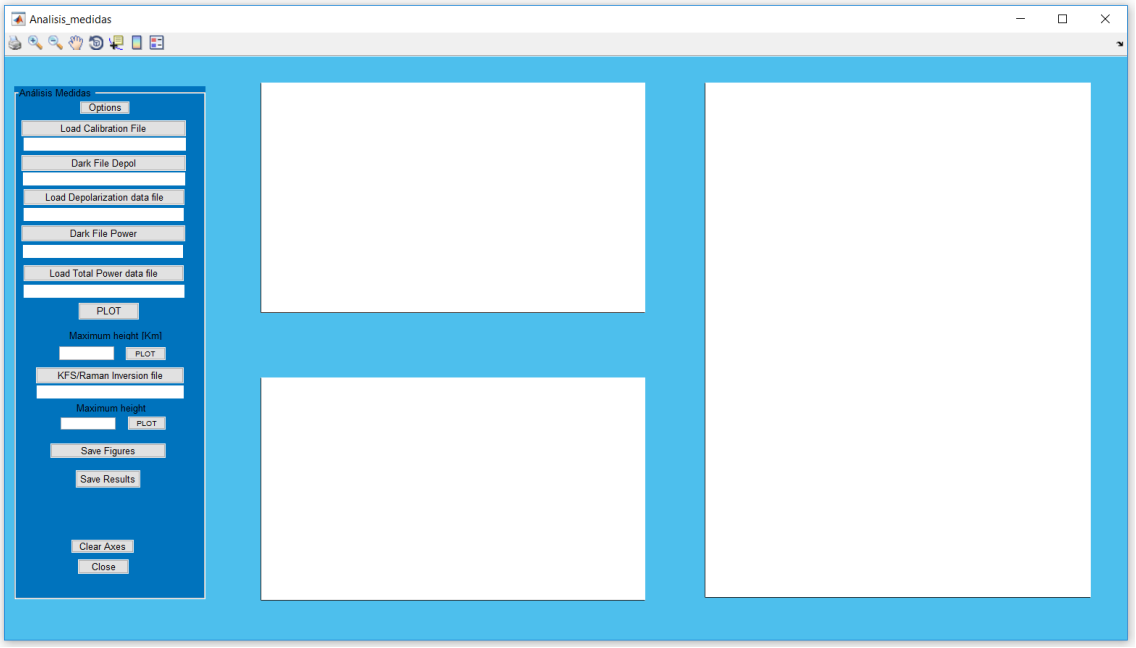

Fig. 4.45 GUI de Análisis de medidas.

4.5.1. Diagrama de flujo

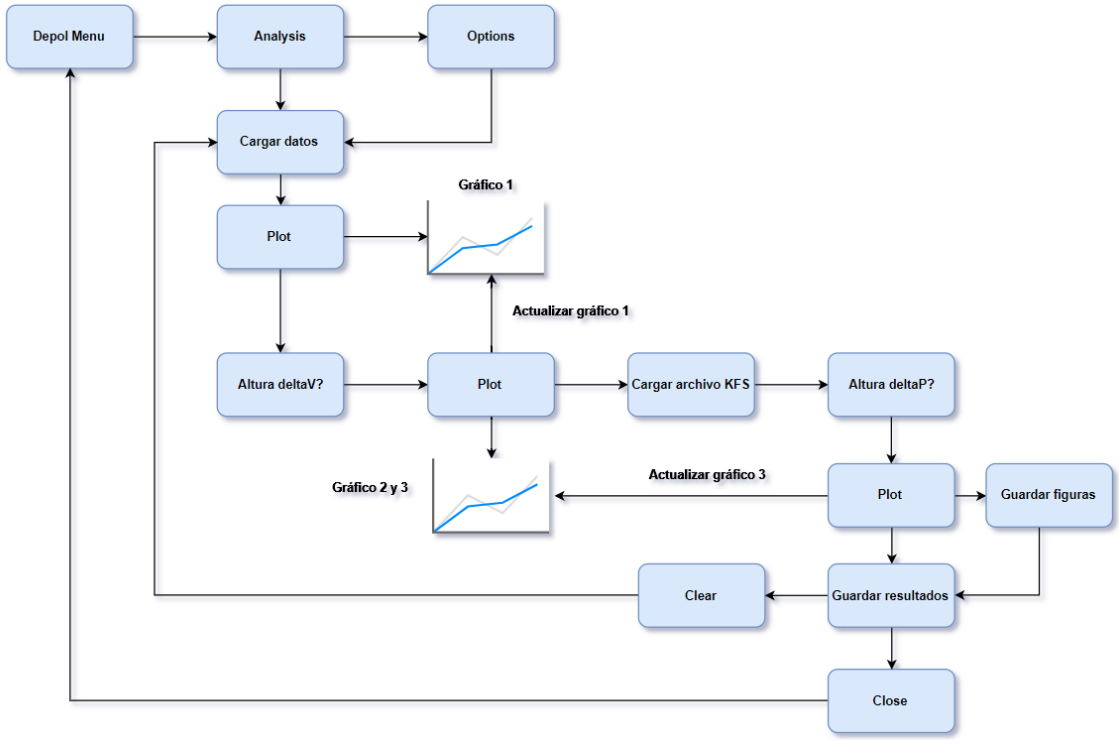

Fig. 4.46 Diagrama de flujo *Analysis.*

4.5.2. Variables iniciales Análisis de medidas (OpeningFcn)

Las variables de inicio que se cargan por defecto en la GUI de análisis de medidas, son muy similares a las explicadas en la GUI de calibración. Por ello, únicamente se explicarán con detalle aquellas variables que no se hayan visto hasta el momento.

Lo primero que se encuentra en esta función, es una línea de código [34] que borra las variables cargadas en el workspace de Matlab.

```
% Borramos variables workspace
evalin( 'base', 'clear variables')
% Guardamos en handles.W el directorio donde se encuentra la GUI
W=what;
handles.W=W.path;
addpath(handles.W); %Añadimos directorio al path de Matlab
```

Fig. 4.47 Código para borrar variables workspace.

Borrar el contenido del workspace cuando se inicia la GUI es muy importante debido a que muchas variables son idénticas para la GUI de calibración y de análisis de medidas. Por este motivo, se borran las variables y después se cargan las variables que se utilizan en esta GUI.

```
global bin_shift_a bin_shift_ac deltaR deltaM532 deltaM355 dire B S dark_vector dark_vector1

% Iniciamos variables globales con valores por defecto
bin_shift_a=25; %samples
assignin('base','bin_shift_a',bin_shift_a)
bin_shift_ac=28; %samples
assignin('base','bin_shift_ac',bin_shift_ac)
deltaR=3.75e-3; %Km
assignin('base','deltaR',deltaR)
deltaM532=3.8e-3;
assignin('base','deltaM532',deltaM532)
deltaM355=8e-3;
assignin('base','deltaM355',deltaM355)

% Asignamos directorio a variable global
dire=handles.W;
assignin('base','dire',dire)

% Variables de selección de archivo
B='m';
S=1;
```

Fig. 4.48 Variables por defecto en OpeningFcn.

Entre las variables utilizadas, se puede observar la variable *dire* para movernos entre carpetas y las variables de selección de archivos *B* y *S*, que se ha explicado en el apartado 4.4.3 (ver Fig. 4.18).

Por último, se encuentran las variables que contienen los archivos de oscuridad para el canal de despolarización y de potencia total. Estos archivos son opcionales, es decir, depende del usuario si son cargados o no. Por defecto, estas variables no son cargadas por el programa y contienen una matriz de ceros

```
% Archivos oscuridad por defecto
dark_vector=zeros(8190,1);
assignin('base','dark_vector',dark_vector)
set(handles.dark_depol_file, 'String', 'Not loaded')
dark_vector1=zeros(8190,1);
assignin('base','dark_vector1',dark_vector1)
set(handles.dark_power_file, 'String', 'Not loaded')
```

Fig. 4.49 Variables oscuridad.

4.5.3. Opciones

Si se pulsa el botón “Options” se pueden ver las variables y el tipo de archivo cargados por defecto en la función *OpeningFcn*. Estos valores pueden ser modificados por el usuario dependiendo de las características del sistema y del tipo de archivo que se va a analizar.

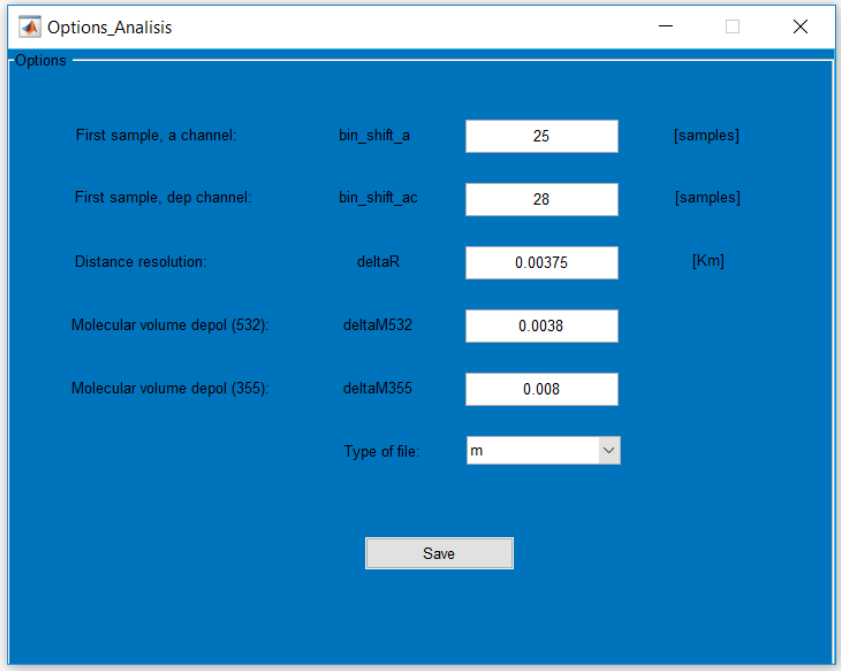

Fig. 4.50 Opciones para análisis de medidas.

4.5.4. Botones de selección de datos

Para empezar a utilizar la GUI de análisis de medidas hay que cargar lo archivos necesarios a través de los cinco botones de la figura 4.51. Con el primer botón ("Load Calibration File) se carga el archivo de calibración que contiene las variables obtenidas a través del proceso de calibración (ver Fig. 4.42).

Los botones "Dark File Depol" y "Dark File Power" se utilizan para cargar las medidas de oscuridad (tipo de fichero .m). Estos archivos son necesarios, especialmente, cuando se trabaje con medidas de 355 nm. Si se decide no cargar estos archivos las variables utilizadas en estas funciones contienen una matriz de ceros (ver Fig. 4.49).

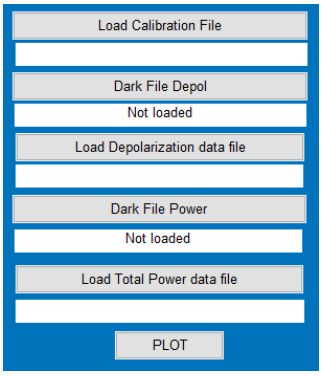

Fig. 4.51 Botones de selección de archivos.

Por defecto, no se carga ningún archivo de oscuridad para realizar el análisis de medidas, pero si son necesarios se pulsaría el botón y se cargaría el archivo correspondiente, teniendo en cuenta la longitud de onda (355 o 532 nm) y el tipo de dato: tipo ac para el archivo de oscuridad de despolarización y tipo a para el archivo de oscuridad de potencia total.

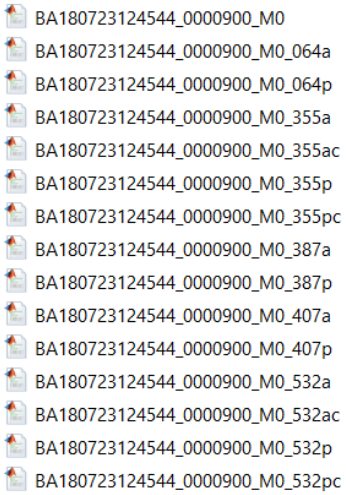

Fig. 4.52 Archivos de oscuridad.

Para los archivos de despolarización y potencia total se pueden cargar dos tipos de datos (fichero .m o fichero .mat), esta elección se realiza con el botón “Options” (ver Fig. 4.50) donde por defecto se cargan los archivos de tipo .m, tal y como se ha programado en el *OpeningFcn* de esta GUI.

Si se decide trabajar con el tipo de archivo cargado por defecto, hay que saber diferenciar que archivo cargará cada botón. Una vez pulsado el botón para cargar el archivo de despolarización o de potencia total nos situamos en la carpeta donde se encuentre la medida y seleccionamos la carpeta “Raw files”. En esta carpeta se encuentran los archivos diferenciados por la longitud de onda (355 o 532 nm) y el tipo de dato, donde los acabados en un solo dígito (a,g,p) corresponden a los archivos de potencia total y los acabados en dos dígitos (ac,gc,pc) corresponden a los archivos de despolarización.

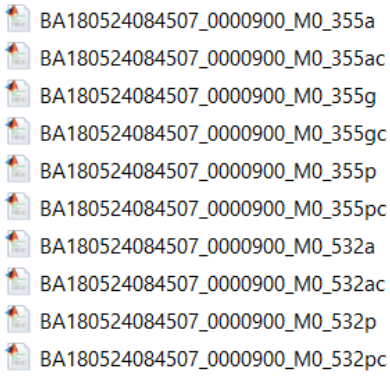

Fig. 4.53 Archivos de despolarización y potencia total.

Si el usuario decide trabajar con archivos tipo .mat, la carpeta donde se encuentran los archivos es diferente. En este caso, la carpeta destino es “Calibrated files” y contiene dos carpetas: “Depol_A” con los archivos de despolarización para las dos longitudes de onda y la carpeta “Total_A” con los archivos de potencia total para las dos longitudes de onda.

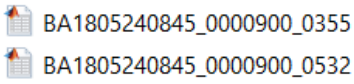

Fig. 4.54 Archivos en carpeta Depol_A.

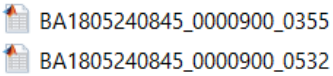

Fig. 4.55 Archivos en carpeta Total_A.

Finalmente, cuando se han seleccionado todos los archivos se muestra el nombre del archivo seleccionado gracias a los textos estáticos que hay debajo de cada botón.

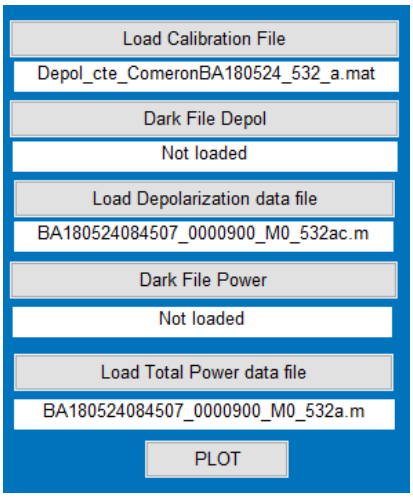

Fig. 4.56 Archivos seleccionados.

4.5.5. Botón Plot

Una vez seleccionados los archivos, se pulsa el botón “PLOT” para mostrar por pantalla los primeros resultados. Los datos se representan en la gráfica superior de la GUI (ver Fig. 4.45).

En la figura 4.57, se muestra en color azul señal del canal de despolarización y en color verde la señal potencia total recibida. Estas señales muestran datos

hasta una altura de 30,71 km, con presencia creciente de ruido como efecto de la multiplicación por el cuadrado de la altura, procedimiento necesario para poder apreciar los detalles de la señal eco [17].

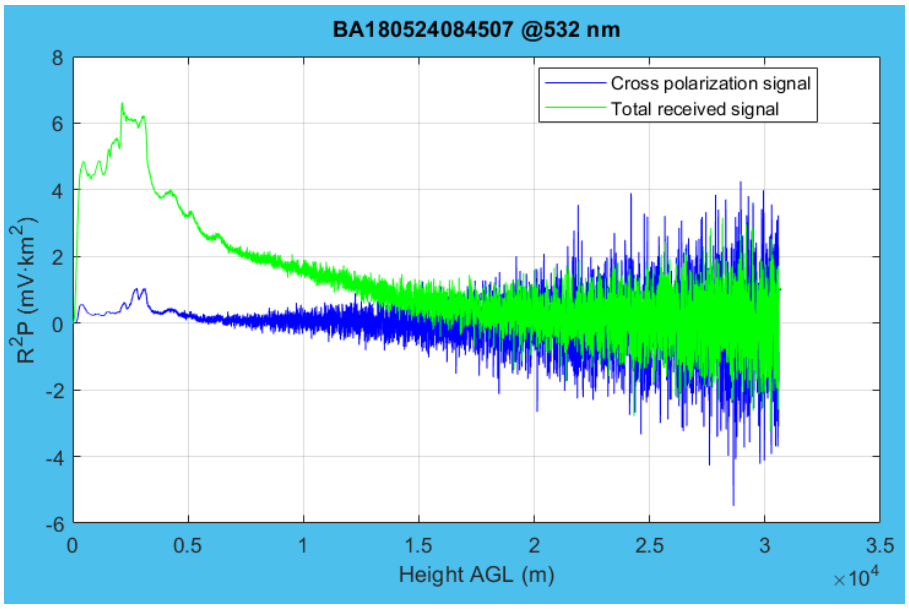

Fig. 4.57 Gráfica señal despolarización y potencia total.

4.5.6. Altura máxima para despolarización de volumen

Con los resultados obtenidos en la figura 4.57, hay que establecer una altura máxima para poder seguir trabajando con la aplicación.

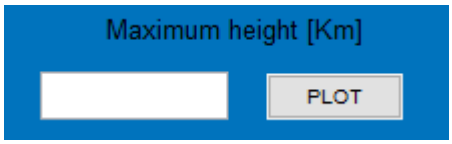

Fig. 4.58 Altura para despolarización de volumen.

Para los resultados que se mostrarán a continuación se ha seleccionado una altura máxima de 10 Km.

Primero de todo, la gráfica con las señales del canal de despolarización y de potencia total recibida (ver Fig. 4.57) se actualiza en función de la altura máxima escogida, mostrando los datos comprendidos en el rango seleccionado:

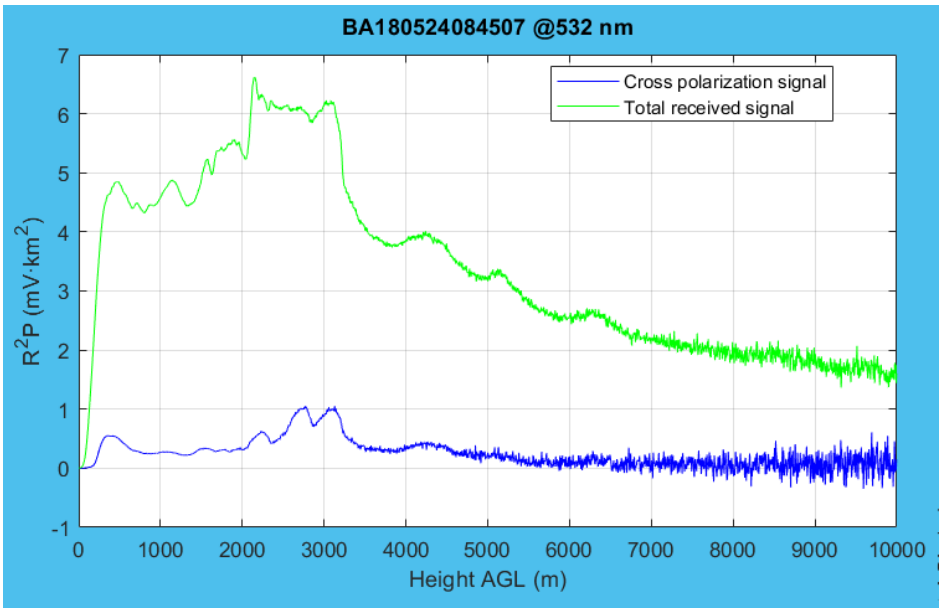

Fig. 4.59 Gráfica señal despolarización y potencia total actualizada.

En la figura 4.60, se muestra el observable $\delta^*(\varphi, R)$ obtenido a partir de la ecuación **1.9**.

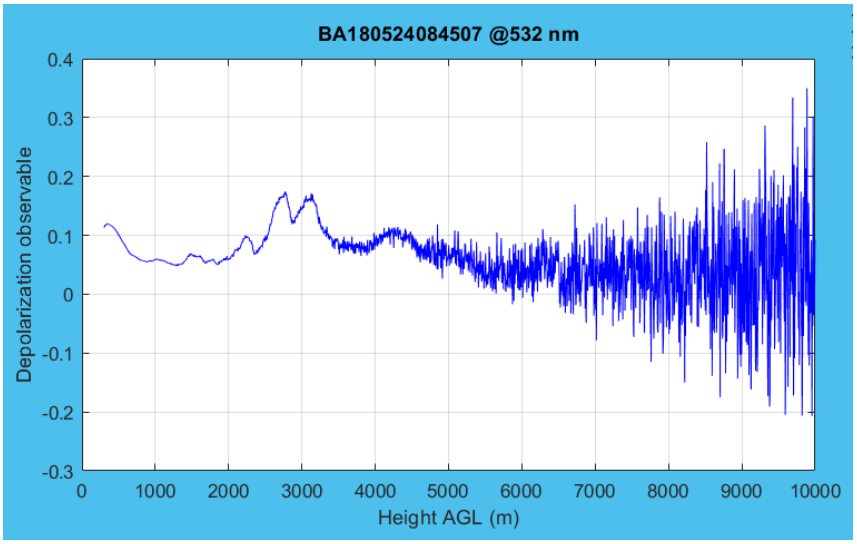

Fig. 4.60 Gráfica del observable.

El último gráfico muestra, en color verde, la despolarización de volumen $\delta^V(R)$ calculada a partir de la ecuación **1.17**. Por otro lado, en color rojo, se muestra la despolarización de volumen teórica, debido a los gases de la atmósfera, que da como resultado un valor $3.8 \cdot 10^{-3}$ calculado por Behrendt y Nakamura [8] para el canal de 532 nm y $8 \cdot 10^{-3}$ calculado por Alejandro Rodríguez [4] para el canal de 355 nm.

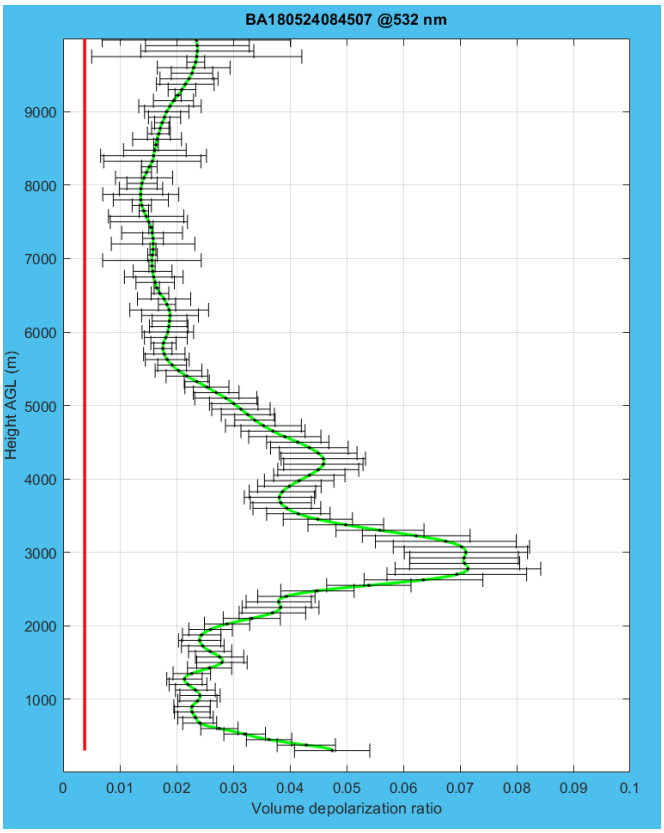

Fig. 4.61 Gráfica despolarización de volumen.

4.5.7. Cargar archivo KFS/Raman

El siguiente paso es cargar el archivo KFS o Raman, ecuación **(1.19)**, y para ello se utiliza el botón de la figura 4.62.

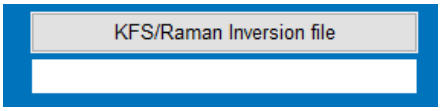

Fig. 4.62 Botón cargar archivo KFS/Raman.

Una vez pulsado el botón nos sitúa en la carpeta de la medida, entramos en la carpeta “Inversion files” y después KFS o Raman. Dentro de estas carpetas se encuentra el archivo para la longitud de onda de 355 nm y 532 nm.

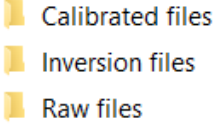

Fig. 4.63 Carpetas de la medida.

Una vez seleccionado el archivo correspondiente, se muestra el nombre gracias al texto estático que se encuentra debajo del botón.

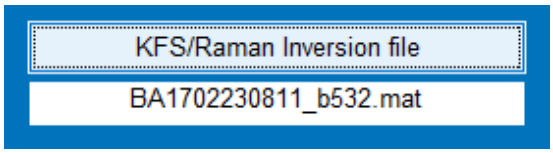

Fig. 4.64 Archivo KFS seleccionado.

Este fichero contiene las variables $\beta^m(R)$ y $\beta^p(R)$ necesarias para calcular el factor $\rho(R)$ de la ecuación **1.19**.

El gráfico de la figura 4.61 se actualiza dibujando, en color azul, la despolarización de partícula $\delta^p(R)$ calculada a partir de la ecuación **1.18**.

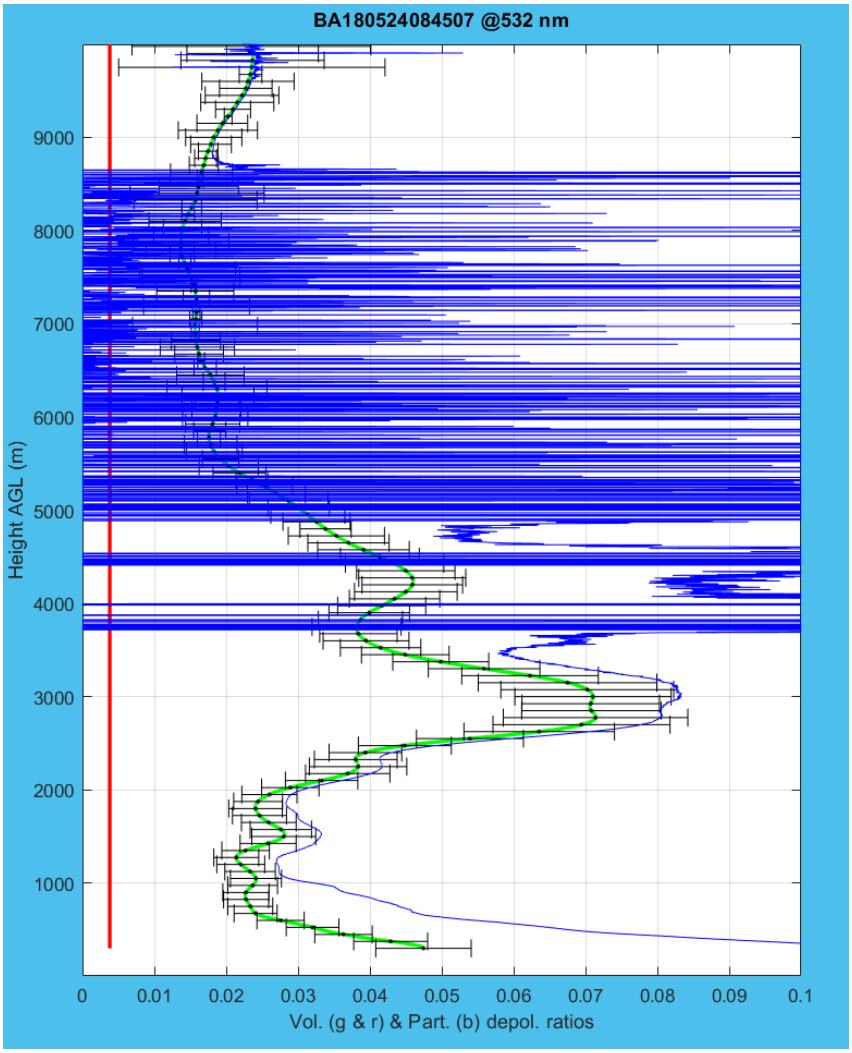

Fig. 4.65 Gráfica despolarización de volumen y partícula.

4.5.8. Altura máxima para despolarización de partícula

El último dato que se introduce en la GUI es la altura máxima para calcular la despolarización de partícula. La altura introducida debe ser menor a la altura introducida para calcular la despolarización de volumen, debido a que el archivo KFS cargado en el apartado anterior solo contiene datos hasta una determinada altura, que es inferior a la introducida para calcular la despolarización de volumen.

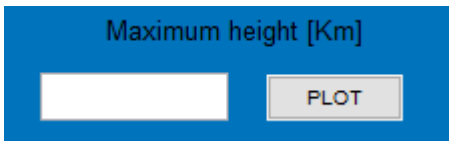

Fig. 4.66 Altura para despolarización de partícula.

Una vez introducida la altura el gráfico de la figura 4.65 se actualiza con la nueva despolarización de partícula.

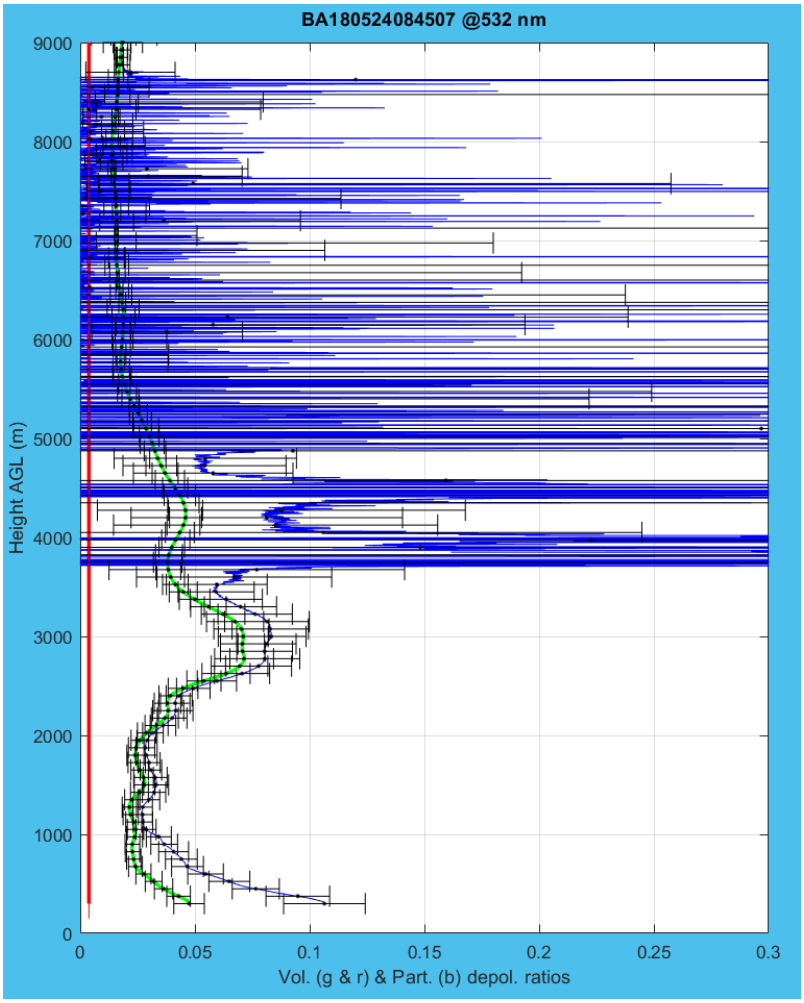

Fig. 4.67 Gráfica despolarización de volumen y partícula actualizada.

4.5.9. Guardar gráficos obtenidos

Para guardar los resultados obtenidos en las gráficas se utiliza el botón “Save Figures”. Una vez pulsado el botón se le pide al usuario, con la ayuda de la función *uigetdir* [32], que elija la carpeta donde se guardarán las figuras.

Una vez elegida la carpeta y con la función *subplot* [33], se genera una figura con los gráficos obtenidos durante la ejecución de la interfaz.

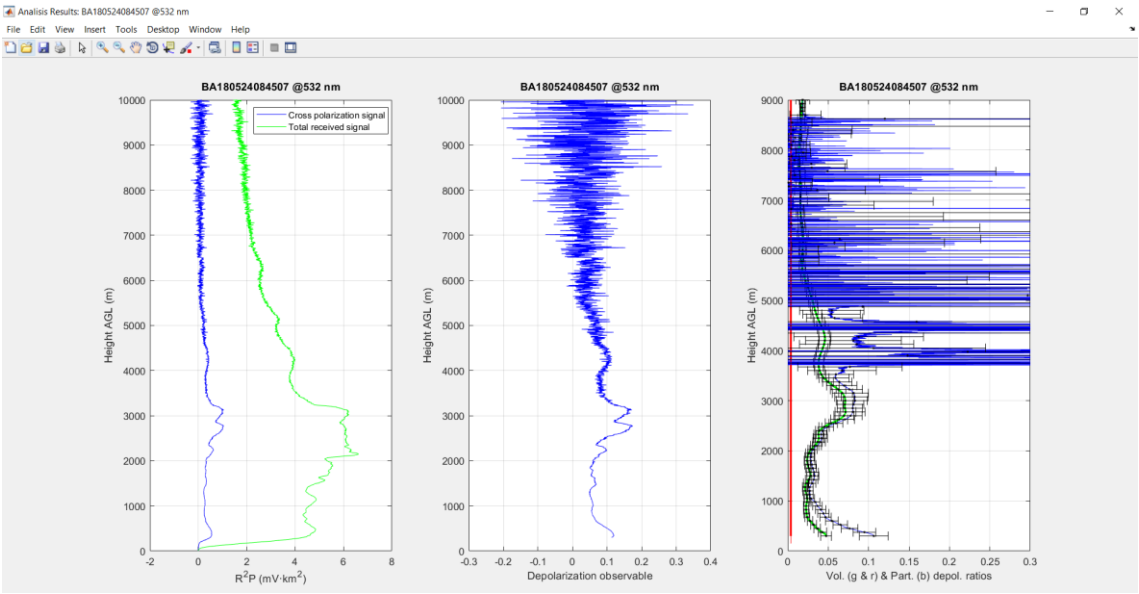

Fig. 4.68 Figura generada con subplot.

```
savedir = uigetdir(dire,'Choose folder to save the figures');
switch B

    case 'm'

    cd(savedir)
    saveas(h1,[name,'_',longonda, '_depol'],'fig')
    saveas(h1,[name,'_',longonda, '_depol'],'jpg')
    saveas(h1,[name,'_',longonda, '_depol'],'tif')

    case 'mat'

    cd(savedir)
    saveas(h1,[name,'_',longonda, '_depol'],'fig')
    saveas(h1,[name,'_',longonda, '_depol'],'jpg')
    saveas(h1,[name,'_',longonda, '_depol'],'tif')

end
```

Fig. 4.69 Código para guardar figuras.

Las figuras obtenidas se guardan en tres formatos distintos, como en la GUI de calibración, que contienen el nombre de la medida (BA1805240845) y la longitud

de onda (532) que se ha seleccionado y utilizado durante la ejecución del programa.

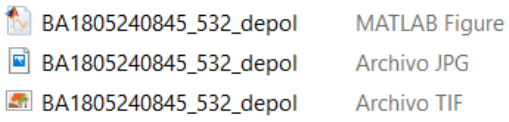

Fig. 4.70 Archivos generados al guardar figuras.

4.5.10. Guardar resultados del análisis de medidas

Para guardar los resultados obtenidos durante la ejecución de la GUI se pulsa el botón “Save Results”, que permite guardar las variables más importantes obtenidas con los cálculos internos del programa.

```
savedir = uigetdir(dire,'Choose folder to save analysis results');
switch B

    case 'm'

fileout=[file_totalp(1:end-17), '_depol_', wavel];
cd(savedir)
save(fileout,'rc','depol_ratio_V_f','depol_ratio_V_Comeron_f','err_depol_ratio_V','depolP','err_depolP','betamie','wavel')

    case 'mat'

fileout=[file_totalp(1:end-17), '_depolmat', wavel];
cd(savedir)
save(fileout,'rc','depol_ratio_V_f','depol_ratio_V_Comeron_f','err_depol_ratio_V','depolP','err_depolP','betamie','wavel')
end
```

Fig. 4.71 Código para guardar archivo del análisis de medidas.

Una vez elegida la ruta donde se guardará el archivo, se genera un archivo .mat con el nombre de la medida y la longitud de onda.

Fig. 4.72 Archivo generado.

El contenido de este archivo se puede ver en la figura 4.73, donde se pueden destacar las variables *depol_ratio_V_Comeron_f* y *depolP*, que corresponden a las variables $\delta^V(R)$ y $\delta^p(R)$.

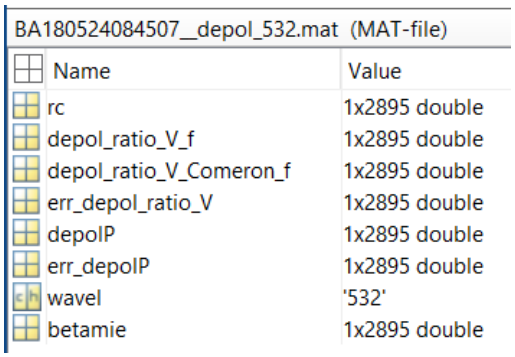

Fig. 4.73 Contenido del archivo generado por el análisis de medidas.

4.5.11. Botones Clear y Close

Por último, si se desea analizar más medidas se pulsaría el botón de “Clear”, con el cual se borraría el contenido de los gráficos de la interfaz gráfica, además de los archivos cargados y valores obtenidos en los cálculos.

Por otro lado, si el usuario desea finalizar la ejecución del programa, presionaría el botón “Close” con el que cerraría la interfaz de análisis de medidas volviendo al menú principal del programa (Depol Menu).

4.5.12. Ejemplo análisis de medidas reciente

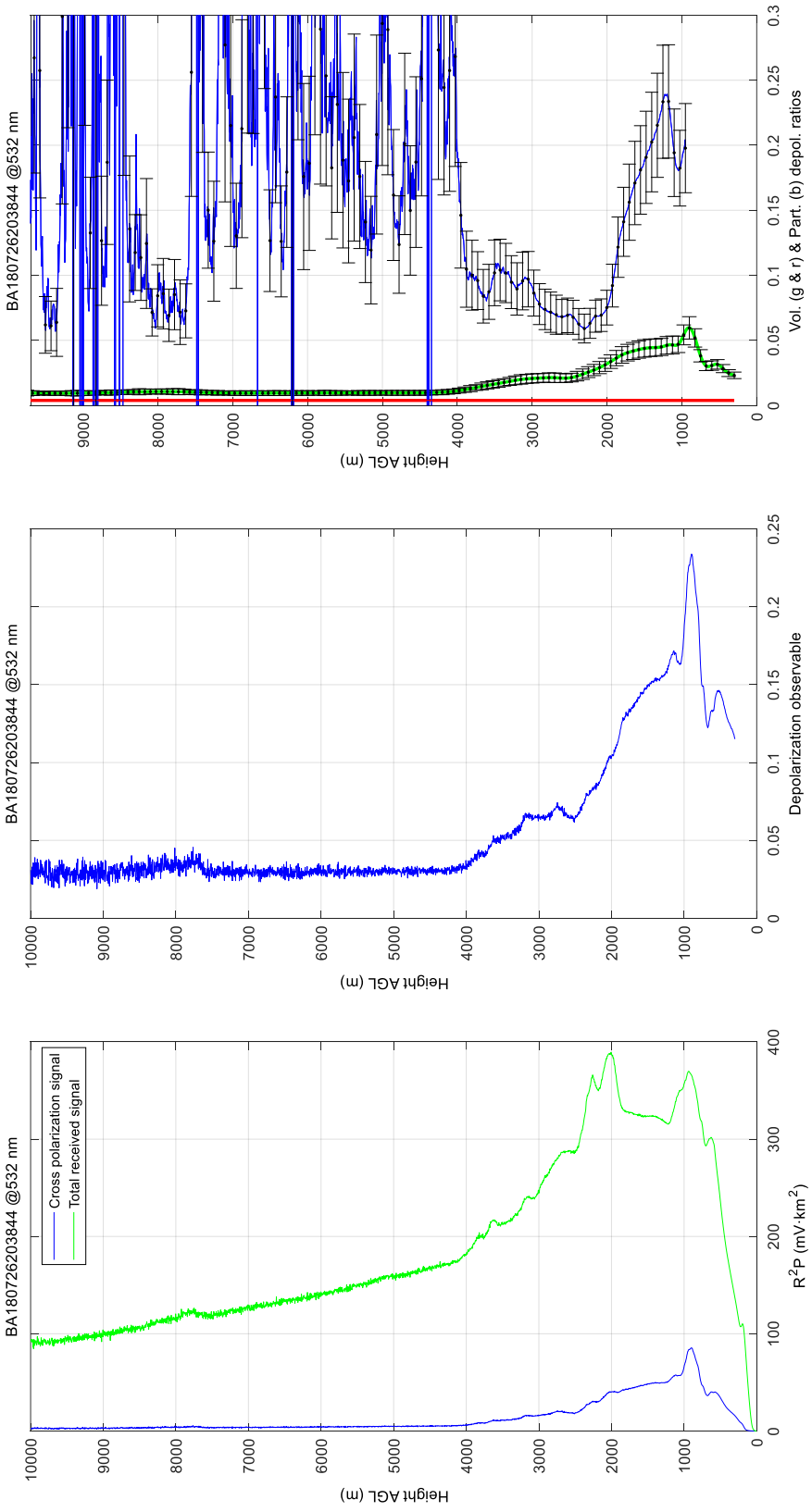

Fig. 4.74 Análisis de medidas 26 de julio de 2018 para 532 nm.

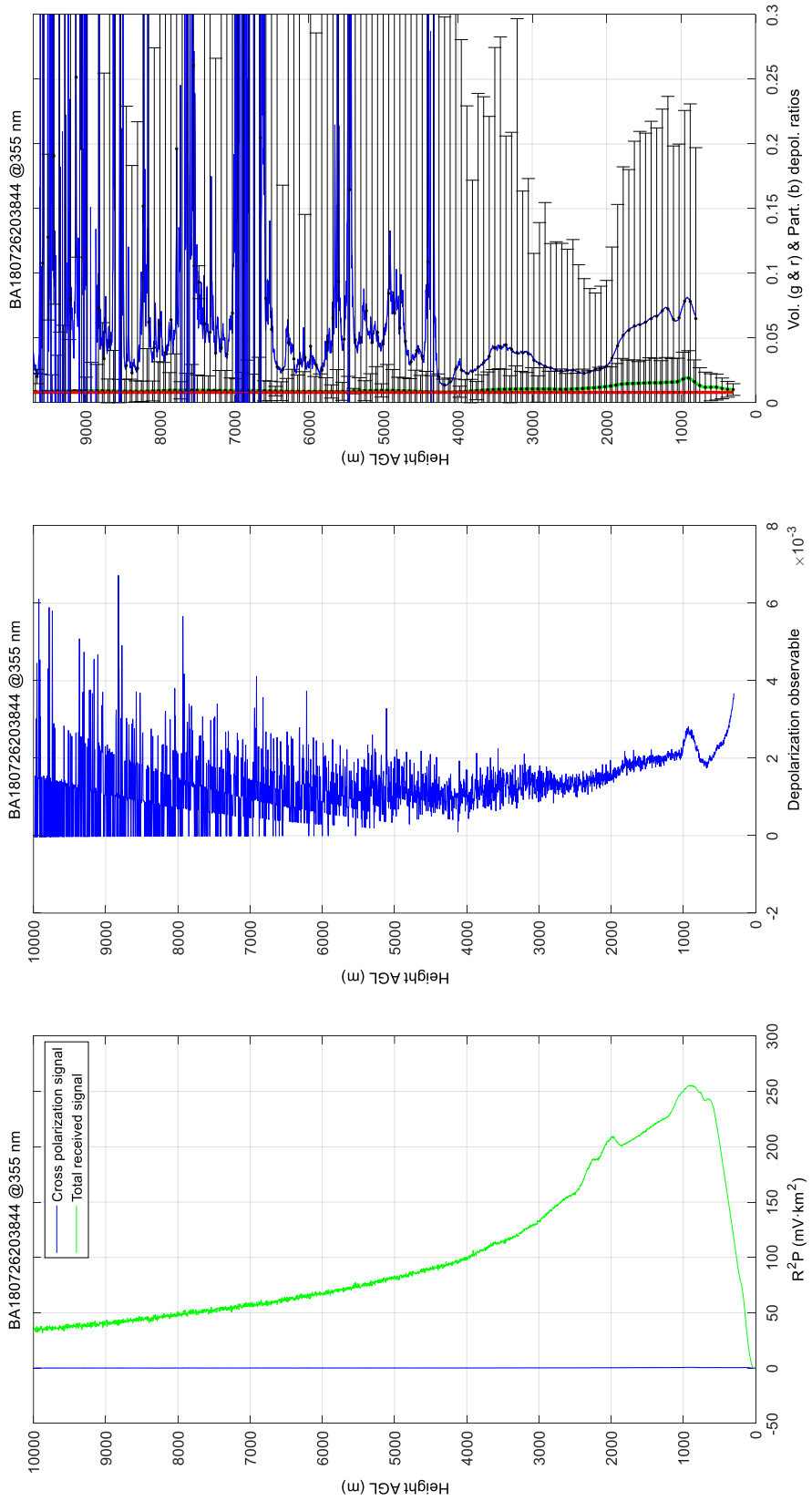

Fig. 4.75 Análisis de medidas 26 de julio de 2018 para 355 nm.

CONCLUSIONES

Una vez finalizado el desarrollo del software y documentado el funcionamiento de las diferentes GUI que forman este proyecto, se concluye que:

1. Se han implementado correctamente todas las funcionalidades que contenían los programas originales.

2. Las funciones internas se comunican correctamente y no da errores.

3. Se han implementado funcionalidades extra, que no estaban presentes en los programas originales.

4. Mediante las opciones disponibles en GUIDE de MATLAB, se han cumplido las necesidades de visualizar datos, elegir archivos e introducir variables.

5. La aplicación desarrollada permite trabajar con dos tipos de datos, dependiendo de las necesidades del usuario.

6. Se ha conseguido que la interfaz gráfica sea robusta, es decir, permite rehacer cálculos sin errores y sin fallos en la ejecución del programa.

7. Los resultados obtenidos, ya sean gráficos o de archivos generados, son idénticos a los obtenidos con los programas originales.

8. Las GUI tienen espacio suficiente por si se decide añadir nuevas funcionalidades en un futuro.

Por otro lado, después de cumplir los objetivos fijados en un principio, hay aspectos que se podrían mejorar para la interfaz desarrollada.

Uno de ellos, podría ser la notificación de errores a través de la interfaz gráfica, ya que actualmente si hay un error se notifica por la ventana de comandos de MATLAB. Estos errores, ya sean debidos a la incorrecta utilización de la GUI o la introducción de valores de manera errónea, se podrían notificar a través de ventanas emergentes para que el usuario pueda proceder a corregir el error.

Otro punto a tener en cuenta es el orden de ejecución del programa, que sigue un orden descendente, donde todos los botones y cuadros de texto están habilitados por defecto. El problema surge si el usuario decide saltarse el orden de ejecución del programa, cosa que provocaría un error en los cálculos. Por lo tanto, una funcionalidad a tener en cuenta es deshabilitar los botones y cuadros de texto que no se utilizan, hasta que llegue el momento de darles uso.

Como conclusión personal, la realización de este trabajo me ha permitido ampliar mis conocimientos en MATLAB y sobretodo aprender a utilizar GUIDE, unas de las herramientas integradas en MATLAB y que permite crear interfaces gráficas de usuario (GUI) de manera sencilla y rápida.

REFERENCIAS Y BIBLIOGRAFÍA

[1] Raymond Measures, *Laser remote sensing Fundamentals and applications,* Krieger, 1992

[2] Francesc Rocadenbosch, Lidar sensing of the atmosphere: Receiver design and inversion algorithms for an elastic system, Universitat Politècnica de Catalunya, 1996.

[3] B. Saleh, M. Teich, *Fundamental of Photonics*, 2nd edition, Wiley 2007.

[4] Alejandro Rodríguez Gómez, Comunicación privada, 2018.

[5] Adolfo Comerón et al, "Considerations about the Determination of the Depolarization Calibration Profile of a Two-Telescope Lidar and Its Implications for Volume Depolarization Ratio Retrieval", *Sensors,* Vol. 18, Issue 6, pp. 1807ss, MDPI 2018, DOI: 10.3390/s18061807 (Open Access).

[6] Alejandro Rodríguez Gómez et al, "An architecture providing depolarization ratio capability for a multi-wavelength raman lidar: Implementation and first measurements", *Sensors,* Vol. 17, Issue 12, MDPI 2017, DOI: 10.3390/s17122957 (Open Access).

[7] Volker Freudenthaler et al, "Depolarization ratio profiling at several wavelengths in pure Saharan dust during SAMUM 2006", *Tellus, Series B: Chemical and Physical Meteorology,* Vol. 61, Issue 1, pp. 165-179, International Meteorological Institute in Stockholm 2009, DOI: 10.1111/j.1600-0889.2008.00396.

[8] Behrendt & Nakamura, "Calculation of the calibration constant of polarization lidar and its dependency on atmospheric temperatura", *Optics Express*, Vol.10, Issue 16, pp. 805ss, OSA, 2002. DOI: 10.1364/OE.10.000805.

[9] "Creación de apps con interfaces gráficas de usuario en MATLAB- MathWorks España." [Online]. Disponible: https://es.mathworks.com/discovery/matlab-gui.html, consultado el 3 de marzo de 2018.

[10] Klett, J.D. Stable analytical inversion solution for processing lidar returns. *Appl Opt.* 1981, 20, 211–220.

[11] Fernald, F.G. Analysis of atmospheric lidar observations: Some comments. *Appl. Opt*. 1984, 23, 652–653.

[12] Ansmann, A.; Riebesell, M.; Weitkamp, C. Measurement of atmospheric aerosol extinction profiles with a Raman lidar. *Opt. Lett*. 1990, 15, 746–748.

[13] Ansmann, A.; Wandinger, U.; Riebesell, M.; Weitkamp, C.; Michaelis, W. Independent measurement of extinction and backscatter profiles in cirrus clouds

by using a combined Raman elastic-backscatter lidar. *Appl. Opt*. 1992, 31, 7113–7131.

[14] Dhiraj Kumar et al, “Six-channel polychromator design and implementation for the UPC elastic/Raman LIDAR”, *SPIE International Symposium - Remote Sensing Europe*, Vol. 8182, pp. 81820W-1-81820W-10, SPIE 2011, DOI: 10.1117/12.896305.

[15] “Interfaz gráfica de usuario - Wikipedia.” [Online]. Disponible: https://es.wikipedia.org/wiki/Interfaz_gr%C3%A1fica_de_usuario, consultado el 27 de febrero de 2018.

[16] Bernd Mielke, *Analog + Photon Counting*, http://licel.com/manuals/analogpc.pdf, consultado el 2 de octubre de 2018.

[17] Alejandro Rodríguez-Gómez et al, “Multi-wavelength aerosol LIDAR signal pre-processing: practical considerations”, *IOP Conference Series: Earth and Environmental Science*, Vol. 28, Issue 1, 2015, http://iopscience.iop.org/article/10.1088/1755-1315/28/1/012013/meta, consultado el 2 de octubre de 2018.

[18] “Guidata - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/guidata.html, consultado el 4 de marzo de 2018.

[19] “Pasar valor de una variables de una GUI a otra.” [Online]. Disponible: https://www.lawebdelprogramador.com/foros/Matlab/1134866-Pasar-valor-de-una-variable-de-una-gui-a-otra.html, consultado el 4 de marzo de 2018.

[20] “Plot - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/plot.html, consultado el 30 de marzo de 2018.

[21] “Plot en axes GUI.” [Online]. Disponible: https://www.lawebdelprogramador.com/foros/Matlab/1301421-Plot-en-axes-GUI.html, consultado el 5 de abril de 2018.

[22] “Imshow - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/images/ref/imshow.html, consultado el 11 de septiembre de 2018.

[23] “Questdlg - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/questdlg.html, consultado el 12 de abril de 2018.

[24] “Strcmp - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/strcmp.html, consultado el 12 de abril de 2018.

[25] “Addpath - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/addpath.html, consultado el 26 de abril de 2018.

[26] “Assignin - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/assignin.html, consultado el 29 de agosto de 2018.

[27] “Set - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/set.html, consultado el 10 de marzo de 2018.

[28] “Get - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/get.html, consultado el 10 de marzo de 2018.

[29] “Switch, case, otherwise - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/switch.html, consultado el 7 de mayo de 2018.

[30] “Uigetfile - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/uigetfile.html, consultado el 16 de mayo de 2018.

[31] “Ylim - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/ylim.html, consultado el 22 de mayo de 2018.

[32] “Uigetdir - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/uigetdir.html, consultado el 17 de septiembre de 2018.

[33] “Subplot - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/help/matlab/ref/subplot.html, consultado el 6 de junio de 2018.

[34] “Clear Global Workspace from GUI Pushbutton - MathWorks España.” [Online]. Disponible: https://es.mathworks.com/matlabcentral/answers/79174-clear-global-workspace-from-gui-pushbutton, consultado el 10 de ocubre de 2018.

[35] Vidal, E. Disseny D’un Canal de Despolarització a 532 nm per al Lidar d’EARLINET de la UPC. BarcelonaTech (2013). Disponible: http://hdl.handle.net/2099.1/18273 , consultado el 19 de julio de 2018.

ANEXOS

ANEXO 1: Código completo Depol Menu

```
function varargout = Depol_Menu(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Depol_Menu_OpeningFcn, ...
                   'gui_OutputFcn',  @Depol_Menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Depol_Menu is made visible.
function Depol_Menu_OpeningFcn(hObject, eventdata, handles, varargin)

axis off
axes(handles.axes1)
imshow('LogoUPC.PNG')

axes(handles.axes2)
imshow('CommSensLab.PNG')

% Choose default command line output for Depol_Menu
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Depol_Menu_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in calibracion.
function calibracion_Callback(hObject, eventdata, handles)

Depol_Calibration_menu;
```

```
close(handles.main);

% --- Executes on button press in analisis_medidas.
function analisis_medidas_Callback(hObject, eventdata, handles)

Depol_Analysis_menu;
close(handles.main);

% --- Executes on button press in Close_main.
function Close_main_Callback(hObject, eventdata, handles)

qt=questdlg(' ¿Do you want to exit from Depol Menu?',...
'Close','Si','No','No');

if strcmp(qt,'No')
     return;
end
evalin( 'base', 'clear variables')
close(handles.main);

% --- Executes when user attempts to close depol_menu.
function main_CloseRequestFcn(hObject, eventdata, handles)

delete(hObject);
```

ANEXO 2: Código completo Calibración

```
function varargout = Depol_Calibration_menu(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Depol_Calibration_menu_OpeningFcn, ...
                   'gui_OutputFcn',  @Depol_Calibration_menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Depol_Calibration_menu is made visible.
function Depol_Calibration_menu_OpeningFcn(hObject, eventdata, handles,
varargin)
% Choose default command line output for Depol_Calibration_menu
```

```
handles.output = hObject;

evalin( 'base', 'clear variables')
% Guardamos en handles.W el directorio donde se encuentra la GUI
W=what;
handles.W=W.path;
addpath(handles.W); %Añadimos directorio al path de Matlab

global Nofilt Dmax deltaR bin_shift_a bin_shift_ac deltaM532 deltaM355 dire S
A

% Iniciamos variables globales con valores por defecto
Nofilt=133; %samples
assignin('base','Nofilt',Nofilt)
Dmax=10; %Km
assignin('base','Dmax',Dmax)
deltaR=3.75e-3; %Km
assignin('base','deltaR',deltaR)
bin_shift_a=25; %samples
assignin('base','bin_shift_a',bin_shift_a)
bin_shift_ac=28; %samples
assignin('base','bin_shift_ac',bin_shift_ac)
deltaM532=3.8e-3;
assignin('base','deltaM532',deltaM532)
deltaM355=8e-3;
assignin('base','deltaM355',deltaM355)

% Asignamos directorio a variable global
dire=handles.W;
assignin('base','dire',dire)

% Variables de selección de archivo
A='m';
S=1;

% Quitamos valores por defecto de los ejes
set(handles.axes1,'xtick',[], 'ytick',[])
set(handles.axes2,'xtick',[], 'ytick',[])
set(handles.axes3,'xtick',[], 'ytick',[])
set(handles.axes4,'xtick',[], 'ytick',[])

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Depol_Calibration_menu_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;

% --- Executes on button press in depolmas45.
function depolmas45_Callback(hObject, eventdata, handles)

global bin_shift_ac r2plus deltaR A dire deltaM532 deltaM355 deltaM wave_mat
```

```
r=deltaR*[1:8189];
r2=r.^2;

cd(dire)
switch A

    case 'm'

[file_desplus,dirinput_desplus]=uigetfile('*.m','Load +45° despolarization
data file');
if isequal(file_desplus,0)
     cd(handles.W)
else
wavel=file_desplus(end-6:end-4);
cd(dirinput_desplus)
dire=dirinput_desplus;

eval([file_desplus(1:end-2)]);
discrim=file_desplus(30);

switch discrim

  case 'a'
        if wavel=='532'
            fileV=[set7a(:,bin_shift_ac+1:end),
set7a(:,length(set7a))*ones(1,bin_shift_ac)];
        elseif wavel=='355'
            fileV=[set8a(:,bin_shift_ac+1:end),
set8a(:,length(set8a))*ones(1,bin_shift_ac)];
        end
    case 'g'
        fileV=set6g;
end
% deltaM se actualiza en función de la longitud de onda
if wavel=='532'

    deltaM=deltaM532;

 elseif wavel=='355'

     deltaM=deltaM355;

end
assignin('base','deltaM',deltaM)

% ----------------------
Nmin=length(fileV)-1001;
Nmax=length(fileV)-1;
% ----------------------
offdesplus=(mean((fileV(:,Nmin:Nmax))'))';
rcorrdp=mean(fileV-offdesplus*ones(1,8190));
r2plus=r2.*rcorrdp(2:end);
assignin('base','r2plus',r2plus)
```

```
%para mostrar archivo selecionado en el static text
filename=file_desplus;
set(handles.filenamedepolmas45,'String',filename);
end
    case 'mat'

cd(dire)
[file_desplus,dirinput_desplus]=uigetfile('*.mat*','Load +45° despolarization
data file');
 if isequal(file_desplus,0)
     cd(handles.W)
else
cd(dirinput_desplus)
dire=dirinput_desplus;

load(file_desplus,'r2p')
r2plus=r2p(2:end);
assignin('base','r2plus',r2plus)
wave_mat=file_desplus(23:25);
assignin('base','wave_mat',wave_mat)

% deltaM se actualiza en función de la longitud de onda
if wave_mat=='532'

    deltaM=deltaM532;

 elseif wave_mat=='355'

     deltaM=deltaM355;
end
assignin('base','deltaM',deltaM)

%para mostrar archivo selecionado en el static text
filename=file_desplus;
set(handles.filenamedepolmas45,'String',filename);

  end
end
cd(handles.W) % volvemos al directorio donde se encuentra la guide

guidata(hObject, handles);

% --- Executes on button press in totpowermas45.
function totpowermas45_Callback(hObject, eventdata, handles)

global bin_shift_a r2totalp deltaR A wavel discrim dire file_totalp

r=deltaR*[1:8189];
r2=r.^2;


cd(dire)
switch A
```

```
    case 'm'

[file_totalp,dirinput_totalp]=uigetfile('*.m','Load total power data file
+45°');
if isequal(file_totalp,0)
     cd(handles.W)
else
cd(dirinput_totalp)
dire=dirinput_totalp;

wavel=file_totalp(end-5:end-3);
assignin('base','wavel',wavel)
eval([file_totalp(1:end-2)]);

discrim=file_totalp(30);

switch discrim

    case 'a'
        if wavel=='532'
            fileV=[set6a(:,bin_shift_a+1:end),
set6a(:,length(set6a))*ones(1,bin_shift_a)];
        elseif wavel=='355'
                fileV=[set1a(:,bin_shift_a+1:end),
set1a(:,length(set1a))*ones(1,bin_shift_a)];
        end
    case 'g'
        fileV=set6g;
end

% ----------------------
Nmin=length(fileV)-1001;
Nmax=length(fileV)-1;
% ----------------------
offtotalp=(mean((fileV(:,Nmin:Nmax))'))';

rcorrtp=mean(fileV-offtotalp*ones(1,8190));
r2totalp=r2.*rcorrtp(2:end);
assignin('base','r2totalp',r2totalp)

%para mostrar archivo selecionado en el static text
filename=file_totalp;
set(handles.powermas45,'String',filename);
end
    case 'mat'
% cd('..')
cd(dire)
[file_totalp,dirinput_totalp]=uigetfile('*.mat*','Load total power data file
+45°');
if isequal(file_totalp,0)
     cd(handles.W)
else
cd(dirinput_totalp)
dire=dirinput_totalp;
```

```
load(file_totalp,'r2p')

r2totalp=r2p(2:end);
assignin('base','r2totalp',r2totalp)

%para mostrar archivo selecionado en el static text
filename=file_totalp;
set(handles.powermas45,'String',filename);

end
end
cd(handles.W) % volvemos al directorio donde se encuentra la guide

guidata(hObject, handles);

% --- Executes on button press in depolmenos45.
function depolmenos45_Callback(hObject, eventdata, handles)

global bin_shift_ac r2pminus deltaR A nombre dire discrim_mat

r=deltaR*[1:8189];
r2=r.^2;

cd(dire)
switch A

    case 'm'
[file_despminus,dirinput_desminus]=uigetfile('*.m','Load -45° despolarization
data file');
if isequal(file_despminus,0)
     cd(handles.W)
else
wavel=file_despminus(end-6:end-4);
cd(dirinput_desminus)
dire=dirinput_desminus;

eval([file_despminus(1:end-2)]);

nombre=file_despminus(1:8);
discrim=file_despminus(30);

switch discrim

    case 'a'
        if wavel=='532'
            fileV=[set7a(:,bin_shift_ac+1:end),
set7a(:,length(set7a))*ones(1,bin_shift_ac)];
        elseif wavel=='355'
            fileV=[set8a(:,bin_shift_ac+1:end),
set8a(:,length(set8a))*ones(1,bin_shift_ac)];
        end
    case 'g'
        fileV=set6g;
end
```

```
% ----------------------
Nmin=length(fileV)-1001;
Nmax=length(fileV)-1;
% ----------------------
offdespminus=(mean((fileV(:,Nmin:Nmax))'))';

rcorrdm=mean(fileV-offdespminus*ones(1,8190));
r2pminus=r2.*rcorrdm(2:end);
assignin('base','r2pminus',r2pminus)

%para mostrar archivo selecionado en el static text
filename=file_despminus;
set(handles.filenamedepolmenos45,'String',filename);
end
    case 'mat'
cd(dire)
[file_despminus,dirinput_desminus]=uigetfile('*.mat*','Load -45°
despolarization data file');
if isequal(file_despminus,0)
     cd(handles.W)
else
cd(dirinput_desminus)
dire=dirinput_desminus;
load(file_despminus,'r2p')

r2pminus=r2p(2:end);
assignin('base','r2pminus',r2pminus)

discrim_mat=dirinput_desminus(end-1);

%para mostrar archivo selecionado en el static text
filename=file_despminus;
set(handles.filenamedepolmenos45,'String',filename);

end
end
cd(handles.W)
guidata(hObject, handles);

% --- Executes on button press in totpowermenos45.
function totpowermenos45_Callback(hObject, eventdata, handles)

global bin_shift_a r2totalm deltaR A dire nombre_mat

r=deltaR*[1:8189];
r2=r.^2;


cd(dire)
switch A

    case 'm'
```

```
[file_totalm,dirinput_totalm]=uigetfile('*.m','Load total power data file -
45°');
if isequal(file_totalm,0)
     cd(handles.W)
else
cd(dirinput_totalm)
dire=dirinput_totalm;

wavel=file_totalm(end-5:end-3); % añadido
eval([file_totalm(1:end-2)]);

discrim=file_totalm(30);

switch discrim

    case 'a'
        if wavel=='532'
            fileV=[set6a(:,bin_shift_a+1:end),
set6a(:,length(set6a))*ones(1,bin_shift_a)];
        elseif wavel=='355'
                fileV=[set1a(:,bin_shift_a+1:end),
set1a(:,length(set1a))*ones(1,bin_shift_a)];
        end
    case 'g'
        fileV=set6g;
end

% ----------------------
Nmin=length(fileV)-1001;
Nmax=length(fileV)-1;
% ----------------------
offtotalm=(mean((fileV(:,Nmin:Nmax))'))';

rcorrtm=mean(fileV-offtotalm*ones(1,8190));
r2totalm=r2.*rcorrtm(2:end);
assignin('base','r2totalm',r2totalm)

%para mostrar archivo selecionado en el static text
filename=file_totalm;
set(handles.powermenos45,'String',filename);
end
    case 'mat'
% cd('..')
cd(dire)
[file_totalm,dirinput_totalm]=uigetfile('*.mat*','Load total power data file -
45°');
if isequal(file_totalm,0)
     cd(handles.W)
else
cd(dirinput_totalm)
dire=dirinput_totalm;
load(file_totalm,'r2p')

r2totalm=r2p(2:end);
```

```
assignin('base','r2totalm',r2totalm)
nombre_mat=file_totalm(1:8);

%para mostrar archivo selecionado en el static text
filename=file_totalm;
set(handles.powermenos45,'String',filename);

end
end
cd(handles.W) % volvemos al directorio donde se encuentra la guide
guidata(hObject, handles);

% --- Executes on button press in plot.
function plot_Callback(hObject, eventdata, handles)

global Dmax deltaR r2plus r2pminus r2totalp r2totalm

Nplot=floor(Dmax/deltaR);

r=deltaR*[1:8189];

% ----------PLOT DEPOL----------
axes(handles.axes1);
plot(r(1:Nplot),r2plus(1:Nplot),r(1:Nplot),r2pminus(1:Nplot))
xlabel('Height AGL (km)')
ylabel('R^2P (mV·km^2)')
lgd=legend('Depolarization signal +45°','Depolarization signal -
45°','Location','Best');
title(lgd,'Depolarization Signal')

% ----------PLOT POWER----------
axes(handles.axes2);
plot(r(1:Nplot),r2totalp(1:Nplot),r(1:Nplot),r2totalm(1:Nplot))
ylim([0 inf])
xlabel('Height AGL (km)')
ylabel('R^2P (mV·km^2)')
lgd=legend('Total received signal +45°','Total received signal -
45°','Location','Best');
title(lgd,'Total Power');

guidata(hObject, handles);

function alturamax_Callback(hObject, eventdata, handles)

global alturamax
altura=get(hObject,'String'); %Almacena valor ingresado
alturamax=str2double(altura); % Transforma a formato double
assignin('base','alturamax',alturamax)

guidata(hObject, handles); % Guarda datos de la aplicacion

% --- Executes during object creation, after setting all properties.
function alturamax_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in calibrar.
function calibrar_Callback(hObject, eventdata, handles)

global deltaM Nofilt alturamax deltaR r2plus r2pminus r2totalp r2totalm
angulo_promedio

r=deltaR*[1:8189];
rmaxi=alturamax;
Nplot=floor(alturamax/deltaR);

ctep=(r2plus./r2totalp);

ctep(1:30)=ctep(30)*ones(1,30);

ctem=(r2pminus./r2totalm);

ctem(1:30)=ctem(30)*ones(1,30);

syscte=(ctep+ctem);

N=length(syscte);

average_interval=find((r<rmaxi));

L=length(average_interval);
System_constant=syscte(average_interval);
System_constant=[System_constant, mean(System_constant(end-100:end))*ones(1,N-
L)];
ctep=[ctep(average_interval), mean(ctep(average_interval(end-
100):average_interval(end)))*ones(1,N-L)];
ctem=[ctem(average_interval), mean(ctem(average_interval(end-
100):average_interval(end)))*ones(1,N-L)];

% Filtering loop:
System_constantf=System_constant(1:Nofilt);
ctep_f=ctep(1:Nofilt);
ctem_f=ctem(1:Nofilt);

for n=Nofilt:average_interval(end)
    pr=ceil((n-Nofilt+1)/2)+1;
    promedio=filtfilt(ones(pr,1),pr,System_constant');
    System_constantf(n)=promedio(n);
    promedio_p=smooth(ctep',pr);
    ctep_f(n)=promedio_p(n);
    promedio_m=smooth(ctem',pr);
    ctem_f(n)=promedio_m(n);
end

System_constantf=[System_constantf, (promedio(n+1:end))'];
```

```
ctep_f=[ctep_f, (promedio_p(n+1:end))'];
ctem_f=[ctem_f, (promedio_m(n+1:end))'];
% -------------------------------------
handles.System_constantf=System_constantf;
handles.ctep_f=ctep_f;
handles.ctem_f=ctem_f;
% -------------------------------------

seno2phi0=(1+deltaM)/(1-deltaM)*(ctem_f-ctep_f)./(ctem_f+ctep_f);
phi0=0.5*(180-asind(seno2phi0));
coordmax=max(System_constantf);

interval=find(((rmaxi-0.5)<r) & (r<rmaxi));
angulo_promedio=mean(phi0(interval));
assignin('base','angulo_promedio',angulo_promedio)
set(handles.angulo,'String',angulo_promedio);
% ------------------------------PLOTS-----------------------------------
% ----------PLOT NEW DEPOL----------
axes(handles.axes1);
plot(r(1:Nplot),r2plus(1:Nplot),r(1:Nplot),r2pminus(1:Nplot))
xlabel('Height AGL (km)')
ylabel('R^2P (mV·km^2)')
lgd=legend('Depolarization signal +45°','Depolarization signal -
45°','Location','Best');
title(lgd,'Depolarization Signal')
% ----------PLOT NEW POWER----------
axes(handles.axes2);
plot(r(1:Nplot),r2totalp(1:Nplot),r(1:Nplot),r2totalm(1:Nplot))
ylim([0 inf])
xlabel('Height AGL (km)')
ylabel('R^2P (mV·km^2)')
lgd=legend('Total received signal +45°','Total received signal -
45°','Location','Best');
title(lgd,'Total Power');
% ----------PLOT CALIBRATION----------
axes(handles.axes3);
plot(r(average_interval),2*ctep_f(average_interval),r(average_interval),2*ctem
_f(average_interval),r(average_interval),System_constantf(average_interval))
hold on
plot(r(average_interval),System_constantf(average_interval),'m','LineWidth',2)
hold off
xlabel('Height AGL (km)')
ylabel('System ratio')
lgd=legend('From +45°','From -45°','Average','Filtered','Location','Best');
title(lgd,'Depolarization Channel System Function')
axis([0 r(average_interval(end)) 0 1.2*coordmax])
% ----------PLOT ANGLE-------
axes(handles.axes4);
plot(r(average_interval),phi0(average_interval));
xlim([rmaxi-0.5 rmaxi])
xlabel('Height AGL (km)')
ylabel('Actual angle (°)')
lgd2=legend('Angle(°)','Location','Best');
title(lgd2,'Actual Polarizer Position')
```

```
guidata(hObject, handles);

% --- Executes on button press in options.
function options_Callback(hObject, eventdata, handles)

Options_Calibration;

uiwait

guidata(hObject, handles);

function plotcalres_Callback(hObject, eventdata, handles)

global Nplot r2plus r2pminus r2totalp r2totalm deltaM Nofilt alturamax deltaR
wavel discrim nombre A dire nombre_mat wave_mat discrim_mat angulo_promedio
file_totalp

savedir = uigetdir(dire,'Choose folder to save the figures');
if isequal(savedir,0)
     cd(handles.W)
else
r=deltaR*[1:8189];

rmaxi=alturamax;
Nplot=floor(alturamax/deltaR);

ctep=(r2plus./r2totalp);

ctep(1:30)=ctep(30)*ones(1,30);

ctem=(r2pminus./r2totalm);

ctem(1:30)=ctem(30)*ones(1,30);

syscte=(ctep+ctem);

N=length(syscte);

average_interval=find((r<rmaxi));

L=length(average_interval);
System_constant=syscte(average_interval);
System_constant=[System_constant, mean(System_constant(end-100:end))*ones(1,N-
L)];
ctep=[ctep(average_interval), mean(ctep(average_interval(end-
100):average_interval(end)))*ones(1,N-L)];
ctem=[ctem(average_interval), mean(ctem(average_interval(end-
100):average_interval(end)))*ones(1,N-L)];

% Filtering loop:
System_constantf=System_constant(1:Nofilt);
ctep_f=ctep(1:Nofilt);
ctem_f=ctem(1:Nofilt);
```

```
for n=Nofilt:average_interval(end)
    pr=ceil((n-Nofilt+1)/2)+1;
    promedio=filtfilt(ones(pr,1),pr,System_constant');
    System_constantf(n)=promedio(n);
    promedio_p=smooth(ctep',pr);
    ctep_f(n)=promedio_p(n);
    promedio_m=smooth(ctem',pr);
    ctem_f(n)=promedio_m(n);
end

System_constantf=[System_constantf, (promedio(n+1:end))'];
ctep_f=[ctep_f, (promedio_p(n+1:end))'];
ctem_f=[ctem_f, (promedio_m(n+1:end))'];

seno2phi0=(1+deltaM)/(1-deltaM)*(ctem_f-ctep_f)./(ctem_f+ctep_f);
phi0=0.5*(180-asind(seno2phi0));
coordmax=max(System_constantf);

% ------------------------------PLOT-------------------------------------
tamano=get(0,'ScreenSize');
titulo=([file_totalp(1:end-18) ' @' wavel ' nm']);
h1=figure('Name',['Calibration Results:
',titulo],'NumberTitle','off','position',[tamano(1) tamano(2) tamano(3)
tamano(4)]);

% ----------PLOT DEPOL----------
subplot(1,3,1)
plot(r2plus(1:Nplot),r(1:Nplot),r2pminus(1:Nplot),r(1:Nplot))
xlabel('R^2P (mV·km^2)')
ylabel('Height AGL (km)')
lgd=legend('Depolarization signal +45°','Depolarization signal -
45°','Location','north');
title(lgd,'Depolarization Signal')

% ----------PLOT POWER----------
subplot(1,3,2)
plot(r2totalp(1:Nplot),r(1:Nplot),r2totalm(1:Nplot),r(1:Nplot))
xlim([0 inf])
xlabel('R^2P (mV·km^2)')
ylabel('Height AGL (km)')
lgd=legend('Total received signal +45°','Total received signal -
45°','Location','north');
title(lgd,'Total Power');

% ----------PLOT CALIBRATION----------
subplot(1,3,3)
plot(2*ctep_f(average_interval),r(average_interval),2*ctem_f(average_interval)
,r(average_interval),System_constantf(average_interval),r(average_interval))
hold on
plot(System_constantf(average_interval),r(average_interval),'m','LineWidth',2)
hold off
xlabel('System ratio')
ylabel('Height AGL (km)')
```

```
angle=num2str(angulo_promedio);
lgd=legend('From +45°','From -45°','Average','Filtered','Location','north');
title(lgd,'Depolarization Channel System Function')
text(0.8*coordmax,0.7*r(average_interval(end)),['Angle: ', angle,'°'])

axis([0 1.2*coordmax 0 r(average_interval(end))])

switch A

    case 'm'
        cd(savedir)
        saveas(h1,[nombre,'_',wavel,'_',  discrim,
'_calibration_results'],'fig')
        saveas(h1,[nombre,'_',wavel,'_',  discrim,
'_calibration_results'],'jpg')
        saveas(h1,[nombre,'_',wavel,'_',  discrim,
'_calibration_results'],'tif')

    case 'mat'
        cd(savedir)
        saveas(h1,[nombre_mat,'_',wave_mat,'_',  discrim_mat,
'_calibration_results'],'fig')
        saveas(h1,[nombre_mat,'_',wave_mat,'_',  discrim_mat,
'_calibration_results'],'jpg')
        saveas(h1,[nombre_mat,'_',wave_mat,'_',  discrim_mat,
'_calibration_results'],'tif')
end
end
cd(handles.W) % volvemos al directorio donde se encuentra la guide

guidata(hObject, handles);

function angulo_Callback(hObject, eventdata, handles)
global angulo

phi0def=get(hObject,'String'); %Almacena valor ingresado
angulo=str2double(phi0def); % Transforma a formato double

guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function angulo_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in savecalibration.
function savecalibration_Callback(hObject, eventdata, handles)

global wavel discrim nombre A dire nombre_mat wave_mat discrim_mat angulo

System_constantf=handles.System_constantf;
```

```
ctep_f=handles.ctep_f;
ctem_f=handles.ctem_f;

angulo=str2double(get(handles.angulo,'String'));
assignin('base','angulo',angulo)
phi0def=angulo;

savedir = uigetdir(dire,'Choose folder to save Calibration');
if isequal(savedir,0)
     cd(handles.W)
else
switch A
case 'm'
    sysfile=['Depol_cte_Comeron', nombre, '_',wavel,'_', discrim];
    cd(savedir)
    eval(['save ', sysfile, ' System_constantf phi0def ctep_f ctem_f']);

case 'mat'

    sysfile=['Depol_cte_Comeron', nombre_mat, '_',wave_mat,'_', discrim_mat];
    cd(savedir)
    eval(['save ', sysfile, ' System_constantf phi0def ctep_f ctem_f']);
end
end
cd(handles.W) % volvemos al directorio donde se encuentra la guide
guidata(hObject, handles);

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)

Depol_Menu;

close(handles.calibration);

function axes1_CreateFcn(hObject, eventdata, handles)
function axes2_CreateFcn(hObject, eventdata, handles)

% --- Executes on button press in clear_axes.
function clear_axes_Callback(hObject, eventdata, handles)

cla(handles.axes1,'reset')
cla(handles.axes2,'reset')
cla(handles.axes3,'reset')
cla(handles.axes4,'reset')

set(handles.filenamedepolmas45, 'String', '')
set(handles.powermas45, 'String', '')
set(handles.filenamedepolmenos45, 'String', '')
set(handles.powermenos45, 'String', '')

set(handles.alturamax, 'String', '')
set(handles.angulo, 'String', '')

evalin( 'base', 'clear variables')
```

```
W=what;
handles.W=W.path;
addpath(handles.W); %Añadimos directorio al path de Matlab

global Nofilt Dmax deltaR bin_shift_a bin_shift_ac deltaM532 deltaM355 dire S
A

% Iniciamos variables globales con valores por defecto
Nofilt=133; %samples
assignin('base','Nofilt',Nofilt)
Dmax=10; %Km
assignin('base','Dmax',Dmax)
deltaR=3.75e-3; %Km
assignin('base','deltaR',deltaR)
bin_shift_a=25; %samples
assignin('base','bin_shift_a',bin_shift_a)
bin_shift_ac=28; %samples
assignin('base','bin_shift_ac',bin_shift_ac)
deltaM532=3.8e-3;
assignin('base','deltaM532',deltaM532)
deltaM355=8e-3;
assignin('base','deltaM355',deltaM355)

% Asignamos directorio a variable global
dire=handles.W;
assignin('base','dire',dire)

% Variables de selección de archivo
A='m';
S=1;

% Quitamos valores por defecto de los ejes
set(handles.axes1,'xtick',[], 'ytick',[])
set(handles.axes2,'xtick',[], 'ytick',[])
set(handles.axes3,'xtick',[], 'ytick',[])
set(handles.axes4,'xtick',[], 'ytick',[])

guidata(hObject, handles);

function calibration_CloseRequestFcn(hObject, eventdata, handles)

delete(hObject);
```

ANEXO 3: Código completo Opciones Calibración

```
function varargout = Options_Calibration(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Options_Calibration_OpeningFcn, ...
```

```
                   'gui_OutputFcn',  @Options_Calibration_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Options_Calibration is made visible.
function Options_Calibration_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for Options_Calibration
handles.output = hObject;

global Nofilt Dmax deltaR bin_shift_a bin_shift_ac deltaM532 deltaM355 S

set(handles.nofilttext,'String',Nofilt) %samples

set(handles.dmaxtext,'String',Dmax) %Km

set(handles.deltartext,'String',deltaR) %Km

set(handles.bin_shift_a_text,'String',bin_shift_a)

set(handles.bin_shift_ac_text,'String',bin_shift_ac)

set(handles.deltam532text,'String',deltaM532)

set(handles.deltam355text,'String',deltaM355)

set(handles.selec_files,'Value',S);

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Options_Calibration_OutputFcn(hObject, eventdata,
handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

function nofilttext_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function nofilttext_CreateFcn(hObject, eventdata, handles)
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function dmaxtext_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function dmaxtext_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function deltartext_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function deltartext_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bin_shift_a_text_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function bin_shift_a_text_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function bin_shift_ac_text_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function bin_shift_ac_text_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function npromintext_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function npromintext_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function deltam532text_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function deltam532text_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in archivo_m.
function archivo_m_Callback(hObject, eventdata, handles)

% --- Executes on button press in archivo_mat.
function archivo_mat_Callback(hObject, eventdata, handles)

% --- Executes on button press in save.
function save_Callback(hObject, eventdata, handles)

global Nofilt Dmax deltaR bin_shift_a bin_shift_ac deltaM532 deltaM355

Nofilt=str2double(get(handles.nofilttext,'String'));
Dmax=str2double(get(handles.dmaxtext,'String'));
deltaR=str2double(get(handles.deltartext,'String'));
bin_shift_a=str2double(get(handles.bin_shift_a_text,'String'));
bin_shift_ac=str2double(get(handles.bin_shift_ac_text,'String'));
deltaM532=str2double(get(handles.deltam532text,'String'));
deltaM355=str2double(get(handles.deltam355text,'String'));

close Options_Calibration

% --- Executes when selected object is changed in selec_archivos.
function selec_archivos_SelectionChangedFcn(hObject, eventdata, handles)

guidata(hObject, handles);

function archivo_mat_CreateFcn(hObject, eventdata, handles)

function deltam355text_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function deltam355text_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in selec_files.
function selec_files_Callback(hObject, eventdata, handles)

global A S
contents = cellstr(get(hObject,'String'));
popChoice = contents(get(hObject,'Value'));
```

```
if (strcmp(popChoice,'m'))
    A='m';
    S=1;
elseif (strcmp(popChoice,'mat'))
    A='mat';
    S=2;
end

% --- Executes during object creation, after setting all properties.
function selec_files_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

ANEXO 4: Código completo Análisis de medidas

```
function varargout = Depol_Analysis_menu(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Depol_Analysis_menu_OpeningFcn, ...
                   'gui_OutputFcn',  @Depol_Analysis_menu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Depol_Analysis_menu is made visible.
function Depol_Analysis_menu_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for Depol_Analysis_menu
handles.output = hObject;
% Borramos variables workspace
evalin( 'base', 'clear variables')
% Guardamos en handles.W el directorio donde se encuentra la GUI
W=what;
handles.W=W.path;
addpath(handles.W); %Añadimos directorio al path de Matlab
```

```
global bin_shift_a bin_shift_ac deltaR deltaM532 deltaM355 dire B S
dark_vector dark_vector1

% Iniciamos variables globales con valores por defecto
bin_shift_a=25; %samples
assignin('base','bin_shift_a',bin_shift_a)
bin_shift_ac=28; %samples
assignin('base','bin_shift_ac',bin_shift_ac)
deltaR=3.75e-3; %Km
assignin('base','deltaR',deltaR)
deltaM532=3.8e-3;
assignin('base','deltaM532',deltaM532)
deltaM355=8e-3;
assignin('base','deltaM355',deltaM355)

% Asignamos directorio a variable global
dire=handles.W;
assignin('base','dire',dire)

% Variables de selección de archivo
B='m';
S=1;

% Archivos oscuridad por defecto
dark_vector=zeros(8190,1);
assignin('base','dark_vector',dark_vector)
set(handles.dark_depol_file, 'String', 'Not loaded')
dark_vector1=zeros(8190,1);
assignin('base','dark_vector1',dark_vector1)
set(handles.dark_power_file, 'String', 'Not loaded')

% Quitamos valores por defecto de los ejes
set(handles.axes1,'xtick',[], 'ytick',[])
set(handles.axes2,'xtick',[], 'ytick',[])
set(handles.axes3,'xtick',[], 'ytick',[])

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Depol_Analysis_menu_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;


% --- Executes on button press in load_calibration_file.
function load_calibration_file_Callback(hObject, eventdata, handles)

[Depol_file,depol_dir]=uigetfile('Depol_cte_Comeron*.mat','Load calibration
file');
if isequal(Depol_file,0)
     cd(handles.W)
else
```

```
cd(depol_dir)

eval(['load ' Depol_file])

load(Depol_file,'System_constantf','phi0def','ctep_f','ctem_f')

handles.System_constantf=System_constantf;
assignin('base','System_constantf',System_constantf)
handles.phi0def=phi0def;
assignin('base','phi0def',phi0def)
handles.ctep_f=ctep_f;
assignin('base','ctep_f',ctep_f)
handles.ctem_f=ctem_f;
assignin('base','ctem_f',ctem_f)
end

filename=Depol_file;
set(handles.Calibration_file,'String',filename);

cd(handles.W) % volvemos al directorio donde se encuentra la guide

guidata(hObject, handles);

% Carga de fichero de oscuridad de despolarización
% --- Executes on button press in dark_file_depol.
function dark_file_depol_Callback(hObject, eventdata, handles)
global dark_vector bin_shift_ac
[file_dark,dir_dark,index_dark]=uigetfile('*.m','Load a dark file for
depolarization channel?');

if index_dark==1
    cd(dir_dark)
    eval(file_dark(1:end-2));
    discrim=file_dark(27:31);
    switch discrim

        case '532ac'
            dark_vector=[set7a(:,bin_shift_ac+1:end),
set7a(:,length(set7a))*ones(1,bin_shift_ac)];
        case '355ac'
            dark_vector=[set8a(:,bin_shift_ac+1:end),
set8a(:,length(set8a))*ones(1,bin_shift_ac)];
    end
            dark_vector=mean(dark_vector,1);
            dark_vector=smooth(dark_vector,80);
else

    dark_vector=zeros(8190,1);

end
assignin('base','dark_vector',dark_vector)

filename=file_dark;
set(handles.dark_depol_file,'String',filename);
```

```
cd(handles.W)

guidata(hObject, handles);
% --- Executes on button press in Load_Depol.
function Load_Depol_Callback(hObject, eventdata, handles)

global dire bin_shift_ac deltaR B name longonda wavel err_rcorrdp r2plus
deltaM532 deltaM355 deltaM dark_vector

rd=deltaR*[1:8189];
r2=rd.^2;

cd(dire)

switch B

    case 'm'

[file_desplus,dirinput_desplus]=uigetfile('*.m','Load despolarization data
file');
if isequal(file_desplus,0)
     cd(handles.W)
else
cd(dirinput_desplus)
dire=dirinput_desplus;

name=file_desplus(1:12);
longonda=file_desplus(27:29);
eval([file_desplus(1:end-2)]);
discrim=file_desplus(27:31);
wavel=discrim(1:3);

switch discrim

    case '532ac'
        fileV=[set7a(:,bin_shift_ac+1:end),
set7a(:,length(set7a))*ones(1,bin_shift_ac)];
    case '355ac'
        fileV=[set8a(:,bin_shift_ac+1:end),
set8a(:,length(set8a))*ones(1,bin_shift_ac)];
    case '532gc'
        fileV=set6g;
    case '355gc'
        fileV=set6g;
end

% deltaM se actualiza en función de la longitud de onda
if wavel=='532'

    deltaM=deltaM532;

 elseif wavel=='355'

     deltaM=deltaM355;
```

```
end
assignin('base','deltaM',deltaM)

% ----------------------
Nmin=length(fileV)-1001;
assignin('base','Nmin',Nmin)
Nmax=length(fileV)-1;
assignin('base','Nmax',Nmax)
% ----------------------

%Sustracción de fichero de oscuridad
tamany=size(fileV);
tamany=tamany(1);
dark_vector=ones(tamany,1)*dark_vector';
fileV=fileV-dark_vector;

% Sustracción offset
offdesplus=(mean((fileV(:,Nmin:Nmax))'))';
rcorrdp=mean(fileV-offdesplus*ones(1,8190));
cd(handles.W) %cambio de directorio para para leer el achivo error_sig
err_rcorrdp=error_sig(rcorrdp(2:end));
assignin('base','err_rcorrdp',err_rcorrdp)
cd(dirinput_desplus)%volvemos al directorio donde estabamos
r2plus=r2.*rcorrdp(2:end);
assignin('base','r2plus',r2plus)

% mostrar archivo en static text
filename=file_desplus;
set(handles.Depol_file,'String',filename);
end

    case 'mat'

    cd(dire)
[file_desplus,dirinput_desplus]=uigetfile('*.mat','Load despolarization data
file');
if isequal(file_desplus,0)
     cd(handles.W)
else
cd(dirinput_desplus)
dire=dirinput_desplus;

name=file_desplus(1:12);
longonda=file_desplus(23:25);
wavel=file_desplus(23:25);

% deltaM se actualiza en función de la longitud de onda
if wavel=='532'

    deltaM=deltaM532;

 elseif wavel=='355'
```

```
     deltaM=deltaM355;

end
assignin('base','deltaM',deltaM)

load(file_desplus,'r2p')
r2plus=r2p(2:end);
assignin('base','r2plus',r2plus)
cd(handles.W) %cambio de directorio para para leer el achivo error_sig
err_rcorrdp=error_sig(r2plus);
assignin('base','err_rcorrdp',err_rcorrdp)
cd(dirinput_desplus)%volvemos al directorio donde estabamos

% mostrar archivo en static text
filename=file_desplus;
set(handles.Depol_file,'String',filename);
end
end

cd(handles.W) % volvemos al directorio donde se encuentra la guide
guidata(hObject, handles);

% Carga de fichero de oscuridad de potencia total
% --- Executes on button press in dark_file_power.
function dark_file_power_Callback(hObject, eventdata, handles)
global dark_vector1 bin_shift_a
[file_dark,dir_dark,index_dark]=uigetfile('*.m','Load a dark file for total
power channel?');

if index_dark==1
    cd(dir_dark)
    eval(file_dark(1:end-2));
    discrim=file_dark(27:31);
    switch discrim

        case '532a'
            dark_vector1=[set6a(:,bin_shift_a+1:end),
set6a(:,length(set6a))*ones(1,bin_shift_a)];
        case '355a'
            dark_vector1=[set1a(:,bin_shift_a+1:end),
set1a(:,length(set1a))*ones(1,bin_shift_a)];
    end
            dark_vector1=mean(dark_vector1,1);
            dark_vector1=smooth(dark_vector1,80);
else
    dark_vector1=zeros(8190,1);
end
assignin('base','dark_vector1',dark_vector1)

filename=file_dark;
set(handles.dark_power_file,'String',filename);

cd(handles.W)
guidata(hObject, handles);
```

```
% --- Executes on button press in Load_totalpower.
function Load_totalpower_Callback(hObject, eventdata, handles)

global dire bin_shift_a deltaR B file_totalp err_rcorrtp dirinput_totalp
r2totalp dark_vector1

rd=deltaR*[1:8189];
r2=rd.^2;

cd(dire)

switch B

    case 'm'
[file_totalp,dirinput_totalp]=uigetfile('*.m','Load total power data file');
if isequal(file_totalp,0)
     cd(handles.W)
else
cd(dirinput_totalp)
dire=dirinput_totalp;

eval([file_totalp(1:end-2)]);
discrim=file_totalp(27:30);

switch discrim

    case '532a'
        fileV=[set6a(:,bin_shift_a+1:end),
set6a(:,length(set6a))*ones(1,bin_shift_a)];
    case '355a'
        fileV=[set1a(:,bin_shift_a+1:end),
set1a(:,length(set1a))*ones(1,bin_shift_a)];
    case '532g'
        fileV=set6g;
    case '355g'
        fileV=set6g;
end
% ----------------------
Nmin=length(fileV)-1001;
assignin('base','Nmin_Power',Nmin)
Nmax=length(fileV)-1;
assignin('base','Nmax_Power',Nmax)
% ----------------------
% Sustracción de fichero de oscuridad
tamany=size(fileV);
tamany=tamany(1);
dark_vector1=ones(tamany,1)*dark_vector1';
fileV=fileV-dark_vector1;

% Sustracción offset
offtotalp=(mean((fileV(:,Nmin:Nmax))'))';
rcorrtp=mean(fileV-offtotalp*ones(1,8190));
cd(handles.W)
```

```
err_rcorrtp=error_sig(rcorrtp(2:end));
assignin('base','err_rcorrtp',err_rcorrtp)
cd(dirinput_totalp)

r2totalp=r2.*rcorrtp(2:end);
assignin('base','r2totalp',r2totalp)

% mostrar archivo en static text
filename=file_totalp;
set(handles.Totalpower_file,'String',filename);
end

    case 'mat'
cd(dire)
[file_totalp,dirinput_totalp]=uigetfile('*.mat','Load total power data file');
if isequal(file_totalp,0)
     cd(handles.W)
else
cd(dirinput_totalp)
dire=dirinput_totalp;

load(file_totalp,'r2p')
r2totalp=r2p(2:end);
assignin('base','r2totalp',r2totalp)

cd(handles.W)
err_rcorrtp=error_sig(r2totalp);
assignin('base','err_rcorrtp',err_rcorrtp)
cd(dirinput_totalp)

% mostrar archivo en static text
filename=file_totalp;
set(handles.Totalpower_file,'String',filename);
end
end

cd(handles.W) % volvemos al directorio donde se encuentra la guide
guidata(hObject, handles);
% Signal presentation
% --- Executes on button press in plot.
function plot_Callback(hObject, eventdata, handles)

global deltaR r2plus r2totalp file_totalp wavel
rd=deltaR*[1:8189];

% ------------------PLOT----------------------
axes(handles.axes1);
plot(1e3*rd,r2plus,'b')
hold on
plot(1e3*rd,r2totalp,'g')
legend('Cross polarization signal','Total received signal','Location','Best')
xlabel('Height AGL (m)')
ylabel('R^2P (mV·km^2)')
title([file_totalp(1:end-17) ' @' wavel ' nm'])
```

```
grid on

guidata(hObject, handles);


function alturamax_Callback(hObject, eventdata, handles)

global rplotmax

alturaa=get(hObject,'String'); %Almacena valor ingresado
rplotmax=str2double(alturaa); % Transforma a formato double
assignin('base','rplotmax',rplotmax)
guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function alturamax_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% Depolarization observable and Volume depolarization computation
% --- Executes on button press in Plot_Altura_max.
function Plot_Altura_max_Callback(hObject, eventdata, handles)
global deltaR deltaM r2plus r2totalp err_rcorrdp err_rcorrtp file_totalp wavel
rplotmax plot_interval depol_ratio_V depol_ratio_V_f depol_ratio_V_Comeron_f
err_depol_ratio_V
rd=deltaR*[1:8189];

System_constantf=handles.System_constantf;
phi0def=handles.phi0def;
ctep_f=handles.ctep_f;
ctem_f=handles.ctem_f;

plot_interval=find((rd>=0.3)&(rd<rplotmax));
assignin('base','plot_interval',plot_interval)
%
delta90=r2plus./r2totalp;
assignin('base','delta90',delta90)
err_delta90=sqrt(((abs(err_rcorrdp))./(abs(r2totalp))).^2+((abs(err_rcorrtp)).
*(abs(r2plus))./((abs(r2totalp)).^2)).^2);
assignin('base','err_delta90',err_delta90)
%
depol_ratio_V=delta90./(System_constantf-delta90);
assignin('base','depol_ratio_V',depol_ratio_V)

depol_ratio_V_f(1:19)=depol_ratio_V(1:19);
assignin('base','depol_ratio_V_f',depol_ratio_V_f)
for nume=20:max(plot_interval)
    Nprom=floor(nume/10+1);
    promedio=filtfilt(ones(1,Nprom),Nprom,depol_ratio_V(nume-
3*Nprom:nume+3*Nprom));
```

```
    depol_ratio_V_f(nume)=promedio(3*Nprom+1);
end

depol_ratio_V_Comeron=(delta90-
(cosd(phi0def)).^2*(ctep_f+ctem_f))./((sind(phi0def)).^2*(ctep_f+ctem_f)-
delta90);
assignin('base','depol_ratio_V_Comeron',depol_ratio_V_Comeron)

depol_ratio_V_Comeron_f=depol_ratio_V_Comeron(1:19);
assignin('base','depol_ratio_V_Comeron_f',depol_ratio_V_Comeron_f)

for nume=20:max(plot_interval)
    Nprom=floor(nume/10+1);
    promedio=filtfilt(ones(1,Nprom),Nprom,depol_ratio_V_Comeron(nume-
3*Nprom:nume+3*Nprom));
    depol_ratio_V_Comeron_f(nume)=promedio(3*Nprom+1);
end

%Error correction:
err_System_constantf=0.1*max(System_constantf);
assignin('base','err_System_constantf',err_System_constantf)
err_depol_ratio_V=sqrt(((abs(System_constantf./((System_constantf-
delta90).^2))).*(abs(err_delta90))).^2+((abs(delta90./((System_constantf-
delta90).^2))).*err_System_constantf).^2);
assignin('base','err_depol_ratio_V',err_depol_ratio_V)
cd(handles.W)

% New Signal presentation with Zoom
Nplot=floor(rplotmax/deltaR);
cla(handles.axes1,'reset')
axes(handles.axes1);
plot(1e3*rd(1:Nplot),r2plus(1:Nplot),'b')
hold on
plot(1e3*rd(1:Nplot),r2totalp(1:Nplot),'g')
legend('Cross polarization signal','Total received signal','Location','Best')
xlabel('Height AGL (m)')
ylabel('R^2P (mV·km^2)')
title([file_totalp(1:end-17) ' @' wavel ' nm'])

grid on

% Depolarization observable
axes(handles.axes2);
plot(1e3*rd(plot_interval),delta90(plot_interval),'b')
grid on
xlabel('Height AGL (m)')
ylabel('Depolarization observable')
title([file_totalp(1:end-17) ' @' wavel ' nm'])

axes(handles.axes3);
plot(depol_ratio_V_Comeron_f(plot_interval),1e3*rd(plot_interval),'Color','g',
'LineWidth',2)
hold on
```

```
herrorbar(depol_ratio_V_Comeron_f(plot_interval(1:20:end)),1e3*rd(plot_interva
l(1:20:end)),err_depol_ratio_V(plot_interval(1:20:end)),'.k')

% Molecular volume depolarization
plot(deltaM*ones(size(plot_interval)),1e3*rd(plot_interval),'Color','r','LineW
idth',2)
axis([0, .1, 1e3*rd(1), 1e3*rd(plot_interval(end))])
grid on
ylabel('Height AGL (m)')
xlabel('Volume depolarization ratio')
title([file_totalp(1:end-17) ' @' wavel ' nm'])

guidata(hObject, handles);

% Particle depolarization computation
% --- Executes on button press in KFS.
function KFS_Callback(hObject, eventdata, handles)

global B deltaR deltaM plot_interval depol_ratio_V_f depol_ratio_V_Comeron_f
err_depol_ratio_V dirinput_totalp file_totalp wavel file_beta532
dirinput_beta532 rc lengthh depolP err_depolP betamie
rd=deltaR*[1:8189];

switch B

    case 'm'

cd(dirinput_totalp)
cd('..')

[file_beta532,dirinput_beta532]=uigetfile('*.mat*','KFS/Raman Inversion
file');
  if isequal(file_beta532,0)
     cd(dirinput_totalp)
else
    cd(dirinput_beta532)

    if file_beta532(end-7:end-4)==['b' wavel]
        load(file_beta532,'rc','betamie','dbmief','betaray');
        betamie=betamie';
        err_betamie=dbmief;
        rc=rc';

    else if file_beta532(end-7:end-4)==['x' wavel]
        load(file_beta532,'r','bmie0b','dbmie0b','bray0');

        betamie=bmie0b;
        err_betamie=dbmie0b(1,:);
        betaray=bray0;
        rc=1e-3*r;
        end
    end
% mostrar archivo en static text
filename=file_beta532;
```

```
set(handles.KFS_File,'String',filename);
  end
    case 'mat'

cd(dirinput_totalp)
cd('..')
cd('..')
[file_beta532,dirinput_beta532]=uigetfile('*.mat*','KFS/Raman Inversion
file');
 if isequal(file_beta532,0)
     cd(dirinput_totalp)
else
    cd(dirinput_beta532)

    if file_beta532(end-7:end-4)==['b' wavel]
        load(file_beta532,'rc','betamie','dbmief','betaray');
        betamie=betamie';
        err_betamie=dbmief;
        rc=rc';

    else if file_beta532(end-7:end-4)==['x' wavel]
        load(file_beta532,'r','bmie0b','dbmie0b','bray0');

        betamie=bmie0b;
        err_betamie=dbmie0b(1,:);
        betaray=bray0;
        rc=1e-3*r;
        end
    end
% mostrar archivo en static text
filename=file_beta532;
set(handles.KFS_File,'String',filename);
 end
end


rho=(betaray+betamie)./betaray;
err_rho=(abs(err_betamie))./betaray;

depol_ratio_V_f=interp1(rd(plot_interval),depol_ratio_V_f(plot_interval),rc);
depol_ratio_V_Comeron_f=interp1(rd(plot_interval),depol_ratio_V_Comeron_f(plot
_interval),rc);
err_depol_ratio_V=interp1(rd(plot_interval),err_depol_ratio_V(plot_interval),r
c);
lengthh=length(rc);

NdepolP=((1+deltaM)*(depol_ratio_V_Comeron_f).*rho-
deltaM*(1+depol_ratio_V_Comeron_f));
DdepolP=((1+deltaM)*rho-(1+depol_ratio_V_Comeron_f));
depolP=NdepolP./DdepolP;

err_depolP=sqrt(((abs((((1+deltaM).*rho-
deltaM).*DdepolP+NdepolP)./(DdepolP.^2))).*abs(err_depol_ratio_V)).^2+((abs(((
```

```
1+deltaM).*depol_ratio_V_Comeron_f.*DdepolP-
(1+deltaM).*NdepolP)./(DdepolP.^2))).*abs(err_rho)).^2);

cd(handles.W) % volvemos al directorio donde se encuentra la guide

% --------------------------------PLOT-----------------
axes(handles.axes3);
plot(depolP,1e3*rc,'b')
grid on
ylabel('Height AGL (m)')
xlabel('Vol. (g & r) & Part. (b) depol. ratios')
title([file_totalp(1:end-17) ' @' wavel ' nm'])

guidata(hObject, handles);

function maxheight_Callback(hObject, eventdata, handles)
global rmax

maxheight=get(hObject,'String'); %Almacena valor ingresado
rmax=str2double(maxheight); % Transforma a formato double

guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function maxheight_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in Plot_Max_Height.
function Plot_Max_Height_Callback(hObject, eventdata, handles)
global rc depol_ratio_V_Comeron_f err_depol_ratio_V depolP err_depolP
file_totalp wavel rmax deltaM

nmax=find(rc>=rmax);
nmax=nmax(1);

axes(handles.axes3);
plot(depol_ratio_V_Comeron_f(1:nmax),1e3*rc(1:nmax),'g')
hold on
plot(depolP(1:nmax),1e3*rc(1:nmax),'b')
plot(deltaM*ones(1,nmax),1e3*rc(1:nmax),'r')
grid on

herrorbar(depol_ratio_V_Comeron_f(1:20:nmax),1e3*rc(1:20:nmax),err_depol_ratio
_V(1:20:nmax),'.k')
herrorbar(depolP(1:20:nmax),1e3*rc(1:20:nmax),err_depolP(1:20:nmax),'.k')

axis([0 0.3 0 1e3*rmax])
ylabel('Height AGL (m)')
xlabel('Vol. (g & r) & Part. (b) depol. ratios')
title([file_totalp(1:end-17) ' @' wavel ' nm'])
```

```
guidata(hObject, handles);


% --- Executes on button press in options_analisis.
function options_analisis_Callback(hObject, eventdata, handles)

Options_Analisis

uiwait

guidata(hObject, handles);


% --- Executes on button press in plot_analisis_results.
function plot_analisis_results_Callback(hObject, eventdata, handles)

global B dire r2plus r2totalp err_rcorrdp err_rcorrtp deltaR deltaM
plot_interval depol_ratio_V_f depol_ratio_V_Comeron_f err_depol_ratio_V
file_totalp wavel rc  depolP err_depolP rmax rplotmax name longonda

savedir = uigetdir(dire,'Choose folder to save the figures');
if isequal(savedir,0)
     cd(handles.W)
else
rd=deltaR*[1:8189];
% r2=rd.^2;

System_constantf=handles.System_constantf;
phi0def=handles.phi0def;
ctep_f=handles.ctep_f;
ctem_f=handles.ctem_f;

% ------------------PLOT SIGNAL PRESENTATION----------------------
tamano=get(0,'ScreenSize');
titulo=([file_totalp(1:end-18) ' @' wavel ' nm']);
h1=figure('Name',['Analisis Results:
',titulo],'NumberTitle','off','position',[tamano(1) tamano(2) tamano(3)
tamano(4)]);
subplot(1,3,1)
Nplot=floor(rplotmax/deltaR);
plot(r2plus(1:Nplot),1e3*rd(1:Nplot),'b')
hold on
plot(r2totalp(1:Nplot),1e3*rd(1:Nplot),'g')
legend('Cross polarization signal','Total received signal','Location','Best')
ylabel('Height AGL (m)')
xlabel('R^2P (mV·km^2)')
title([file_totalp(1:end-17) ' @' wavel ' nm'])
grid on
% ----------------------------------------------------------------

plot_interval=find((rd>=0.3)&(rd<rplotmax));
delta90=r2plus./r2totalp;
```

```
err_delta90=sqrt(((abs(err_rcorrdp))./(abs(r2totalp))).^2+((abs(err_rcorrtp)).
*(abs(r2plus))./((abs(r2totalp)).^2)).^2);

depol_ratio_V=delta90./(System_constantf-delta90);

depol_ratio_V_f(1:19)=depol_ratio_V(1:19);
for nume=20:max(plot_interval)
    Nprom=floor(nume/10+1);
    promedio=filtfilt(ones(1,Nprom),Nprom,depol_ratio_V(nume-
3*Nprom:nume+3*Nprom));
    depol_ratio_V_f(nume)=promedio(3*Nprom+1);
end

depol_ratio_V_Comeron=(delta90-
(cosd(phi0def)).^2*(ctep_f+ctem_f))./((sind(phi0def)).^2*(ctep_f+ctem_f)-
delta90);

depol_ratio_V_Comeron_f=depol_ratio_V_Comeron(1:19);
for nume=20:max(plot_interval)
    Nprom=floor(nume/10+1);
    promedio=filtfilt(ones(1,Nprom),Nprom,depol_ratio_V_Comeron(nume-
3*Nprom:nume+3*Nprom));
    depol_ratio_V_Comeron_f(nume)=promedio(3*Nprom+1);
end

%Error correction:
err_System_constantf=0.1*max(System_constantf);
err_depol_ratio_V=sqrt(((abs(System_constantf./((System_constantf-
delta90).^2))).*(abs(err_delta90))).^2+((abs(delta90./((System_constantf-
delta90).^2))).*err_System_constantf).^2);

cd(handles.W)

% ------------------PLOT DEPOLARIZATION OBSERVABLE----------------------
subplot(1,3,2)
plot(delta90(plot_interval),1e3*rd(plot_interval),'b')
grid on
ylabel('Height AGL (m)')
xlabel('Depolarization observable')
title([file_totalp(1:end-17) ' @' wavel ' nm'])
% ----------------------------------------------------------------------
% Molecular volume depolarization
% ------------------PLOT VOLUME DEPOLARIZATION RATIO----------------------
subplot(1,3,3)
plot(depol_ratio_V_Comeron_f(plot_interval),1e3*rd(plot_interval),'Color','g',
'LineWidth',2)
hold on
herrorbar(depol_ratio_V_Comeron_f(plot_interval(1:20:end)),1e3*rd(plot_interva
l(1:20:end)),err_depol_ratio_V(plot_interval(1:20:end)),'.k')
% Molecular volume depolarization
plot(deltaM*ones(size(plot_interval)),1e3*rd(plot_interval),'Color','r','LineW
idth',2)

axis([0, .1, 1e3*rd(1), 1e3*rd(plot_interval(end))])
```

```
grid on

ylabel('Height AGL (m)')
xlabel('Vol. (g & r) & Part. (b) depol. ratios')

title([file_totalp(1:end-17) ' @' wavel ' nm'])
% --------------------------------------------------------------------
% Particle depolarization computation
cd(handles.W)

subplot(1,3,3)
plot(depolP,1e3*rc,'b')
grid on
ylabel('Height AGL (m)')
xlabel('Particle depolarization ratio')
title([file_totalp(1:end-17) ' @' wavel ' nm'])

nmax=find(rc>=rmax);
nmax=nmax(1);

subplot(1,3,3)
% plot(depol_ratio_V_Comeron_f(1:nmax),1e3*rc(1:nmax),'g')
hold on
plot(depolP(1:nmax),1e3*rc(1:nmax),'b')
plot(deltaM*ones(1,nmax),1e3*rc(1:nmax),'r')
grid on

herrorbar(depolP(1:20:nmax),1e3*rc(1:20:nmax),err_depolP(1:20:nmax),'.k')
axis([0 0.3 0 1e3*rmax])
ylabel('Height AGL (m)')
xlabel('Vol. (g & r) & Part. (b) depol. ratios')
title([file_totalp(1:end-17) ' @' wavel ' nm'])

switch B

    case 'm'

    cd(savedir)
    saveas(h1,[name,'_',longonda, '_depol'],'fig')
    saveas(h1,[name,'_',longonda, '_depol'],'jpg')
    saveas(h1,[name,'_',longonda, '_depol'],'tif')

    case 'mat'

    cd(savedir)
    saveas(h1,[name,'_',longonda, '_depol'],'fig')
    saveas(h1,[name,'_',longonda, '_depol'],'jpg')
    saveas(h1,[name,'_',longonda, '_depol'],'tif')

end
end
cd(handles.W)

guidata(hObject, handles);
```

```
% --- Executes on button press in save_analysis_results.
function save_analysis_results_Callback(hObject, eventdata, handles)

global B file_totalp dirinput_totalp rc depol_ratio_V_f
depol_ratio_V_Comeron_f err_depol_ratio_V depolP err_depolP lengthh
file_beta532 dirinput_beta532 wavel betamie dire

savedir = uigetdir(dire,'Choose folder to save analysis results');
if isequal(savedir,0)
     cd(handles.W)
else
switch B

    case 'm'

fileout=[file_totalp(1:end-17), '_depol_', wavel];
cd(savedir)
save(fileout,'rc','depol_ratio_V_f','depol_ratio_V_Comeron_f','err_depol_ratio
_V','depolP','err_depolP','betamie','wavel')

    case 'mat'

fileout=[file_totalp(1:end-17), '_depolmat', wavel];
cd(savedir)
save(fileout,'rc','depol_ratio_V_f','depol_ratio_V_Comeron_f','err_depol_ratio
_V','depolP','err_depolP','betamie','wavel')
end

%-------------------------------------------
%  NetCDF data output
%-------------------------------------------
cd(dirinput_beta532)

ficheronc=[file_beta532(1:12), '.b', wavel];

% VARIABLES
nccreate(ficheronc,'VolumeDepol','Dimensions',{'Length'
lengthh},'Format','classic','Datatype','double')
ncwrite(ficheronc,'VolumeDepol',(depol_ratio_V_Comeron_f)')

nccreate(ficheronc,'ErrorVolumeDepol','Dimensions',{'Length'
lengthh},'Format','classic','Datatype','double')
ncwrite(ficheronc,'ErrorVolumeDepol',(err_depol_ratio_V)')

nccreate(ficheronc,'ParticleDepol','Dimensions',{'Length'
lengthh},'Format','classic','Datatype','double')
ncwrite(ficheronc,'ParticleDepol',depolP')

nccreate(ficheronc,'ErrorParticleDepol','Dimensions',{'Length'
lengthh},'Format','classic','Datatype','double')
ncwrite(ficheronc,'ErrorParticleDepol',(err_depolP)')
```

```
end
cd(handles.W)

guidata(hObject, handles);


% --- Executes on button press in clear_axes.
function clear_axes_Callback(hObject, eventdata, handles)

cla(handles.axes1,'reset')
cla(handles.axes2,'reset')
cla(handles.axes3,'reset')
set(handles.Calibration_file, 'String', '')
set(handles.dark_depol_file, 'String', '')
set(handles.Depol_file, 'String', '')
set(handles.dark_power_file, 'String', '')
set(handles.Totalpower_file, 'String', '')
set(handles.KFS_File,'String', '');

set(handles.alturamax, 'String', '')
set(handles.maxheight, 'String', '')

evalin( 'base', 'clear variables')

W=what;
handles.W=W.path;
addpath(handles.W); %Añadimos directorio al path de Matlab

global bin_shift_a bin_shift_ac deltaR deltaM532 deltaM355 dire B S
dark_vector dark_vector1
% Iniciamos variables globales con valores por defecto
bin_shift_a=25; %samples
assignin('base','bin_shift_a',bin_shift_a)
bin_shift_ac=28; %samples
assignin('base','bin_shift_ac',bin_shift_ac)
deltaR=3.75e-3; %Km
assignin('base','deltaR',deltaR)
deltaM532=3.8e-3;
assignin('base','deltaM532',deltaM532)
deltaM355=8e-3;
assignin('base','deltaM355',deltaM355)

% Asignamos directorio a variable global
dire=handles.W;
assignin('base','dire',dire)

% Variables de selección de archivo
B='m';
S=1;

% Archivos oscuridad por defecto
dark_vector=zeros(8190,1);
assignin('base','dark_vector',dark_vector)
set(handles.dark_depol_file, 'String', 'Not loaded')
```

```
dark_vector1=zeros(8190,1);
assignin('base','dark_vector1',dark_vector1)
set(handles.dark_power_file, 'String', 'Not loaded')

% Quitamos valores por defecto de los ejes
set(handles.axes1,'xtick',[], 'ytick',[])
set(handles.axes2,'xtick',[], 'ytick',[])
set(handles.axes3,'xtick',[], 'ytick',[])

guidata(hObject, handles);

% --- Executes on button press in close.
function close_Callback(hObject, eventdata, handles)

Depol_Menu;

close(handles.analisis_medidas);
```

ANEXO 5: Código completo Opciones Análisis de medidas

```
function varargout = Options_Analisis(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Options_Analisis_OpeningFcn, ...
                   'gui_OutputFcn',  @Options_Analisis_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Options_Analisis is made visible.
function Options_Analisis_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for Options_Analisis
handles.output = hObject;

global bin_shift_a bin_shift_ac deltaR deltaM532 deltaM355 S

set(handles.binshiftatxt,'String',bin_shift_a)

set(handles.binshiftactxt,'String',bin_shift_ac)
```

```
set(handles.deltartxt,'String',deltaR)

set(handles.deltam532txt,'String',deltaM532)

set(handles.deltam355txt,'String',deltaM355)

set(handles.selec_files,'Value',S);

% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = Options_Analisis_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

function binshiftatxt_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function binshiftatxt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function binshiftactxt_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function binshiftactxt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function deltartxt_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function deltartxt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function deltam532txt_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function deltam532txt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in saveparams.
function saveparams_Callback(hObject, eventdata, handles)

global bin_shift_a bin_shift_ac deltaR deltaM532 deltaM355

bin_shift_a=str2double(get(handles.binshiftatxt,'String'));
assignin('base','bin_shift_a',bin_shift_a)
bin_shift_ac=str2double(get(handles.binshiftactxt,'String'));
assignin('base','bin_shift_ac',bin_shift_ac)
deltaR=str2double(get(handles.deltartxt,'String'));
assignin('base','deltaR',deltaR)
deltaM532=str2double(get(handles.deltam532txt,'String'));
assignin('base','deltaM532',deltaM532)
deltaM355=str2double(get(handles.deltam355txt,'String'));
assignin('base','deltaM355',deltaM355)

close Options_Analisis

function deltam355txt_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function deltam355txt_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in selec_files.
function selec_files_Callback(hObject, eventdata, handles)

global B S
contents = cellstr(get(hObject,'String'));
popChoice = contents(get(hObject,'Value'));
if (strcmp(popChoice,'m'))
    B='m';
    S=1;
elseif (strcmp(popChoice,'mat'))
    B='mat';
    S=2;
end

% --- Executes during object creation, after setting all properties.
function selec_files_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```